



FEATURE STORE

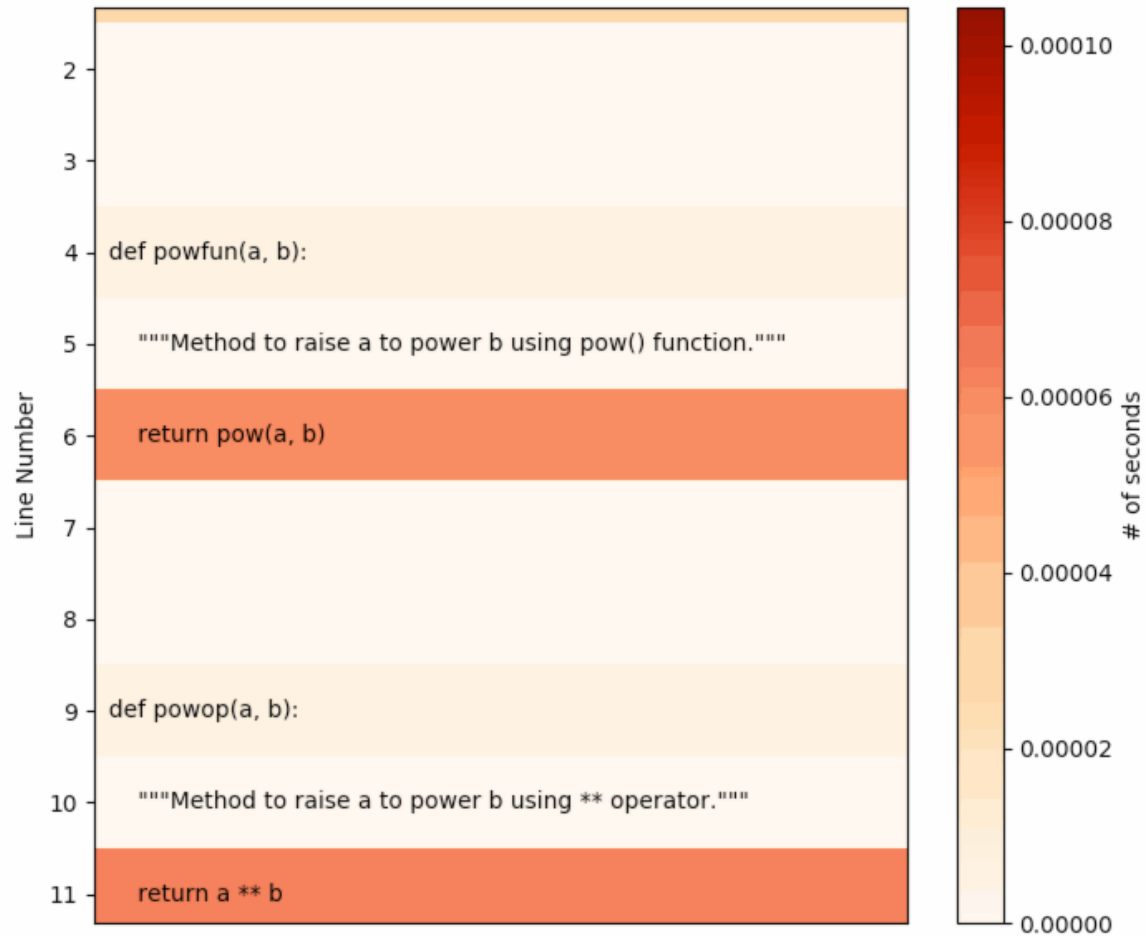
SELEZNEV ARTEM
SENIOR DE @ SBER

HEARTREATE

```
27 1      def merge_sort(m):
28         """
29         Return a sorted copy of m
30         Uses the recursive merge sort algorithm
31         """
32
33 75001    if len(m) <= 1:
34 37500    return m
35 37501    middle = len(m) // 2
36 37501    left = m[:middle]
37 37501    right = m[middle:]
38 37501    return merge(
39 37501    merge_sort(left), merge_sort(right)
40    )
41
42 1      def merge(left, right):
43 37498    result = []
44 37498    left_idx, right_idx = 0, 0
45 518646    while left_idx < len(left) and right_idx < len(right):
46 481149        if left[left_idx] <= right[right_idx]:
47 237261            result.append(left[left_idx])
48 237261            left_idx += 1
49        else:
50 243887            result.append(right[right_idx])
51 243887            right_idx += 1
52 37497    if left_idx < len(left):
53 16843        result.extend(left[left_idx:])
54 37497    if right_idx < len(right):
55 20654        result.extend(right[right_idx:])
56 37497    return result
```

<https://github.com/alexmojaki/hearttrate>

PYHEAT



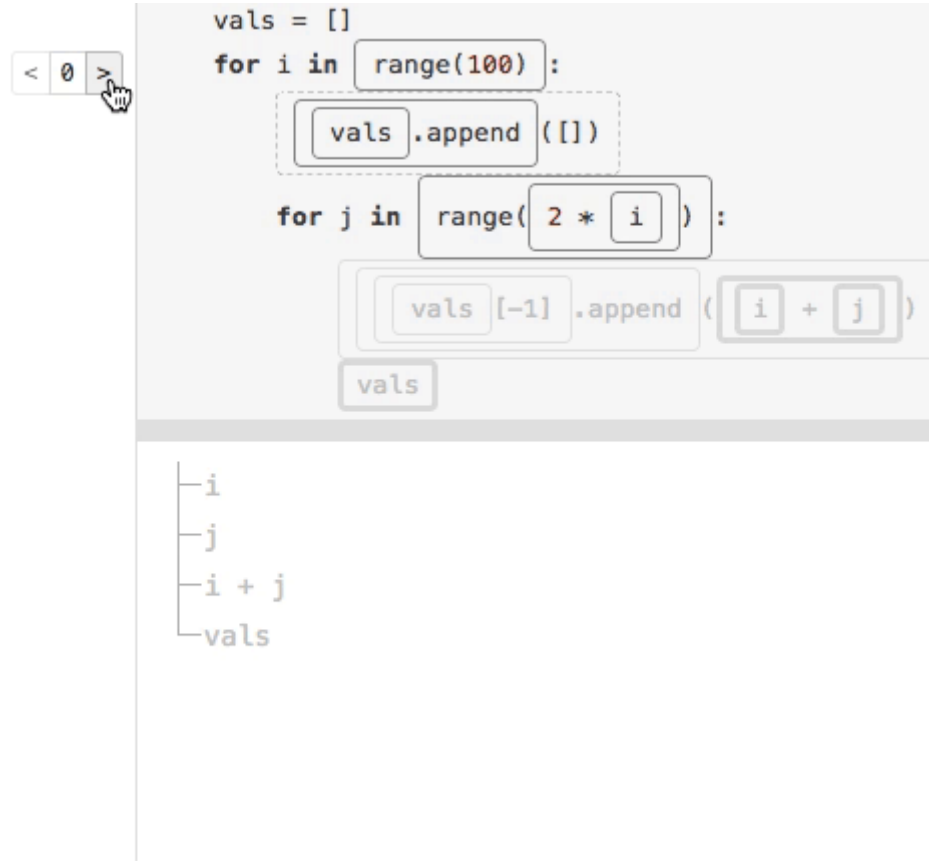
<https://github.com/csurfer/pyheat>

Scroll



10.82

BIRDSEYE



<https://github.com/alexmojaki/birdseye>

НА СЕГОДНЯ

- Фиксация данных
для DS pipeline



great_expectations

НА СЕГОДНЯ

- Фиксация данных
для DS pipeline
- FEATURE STORE

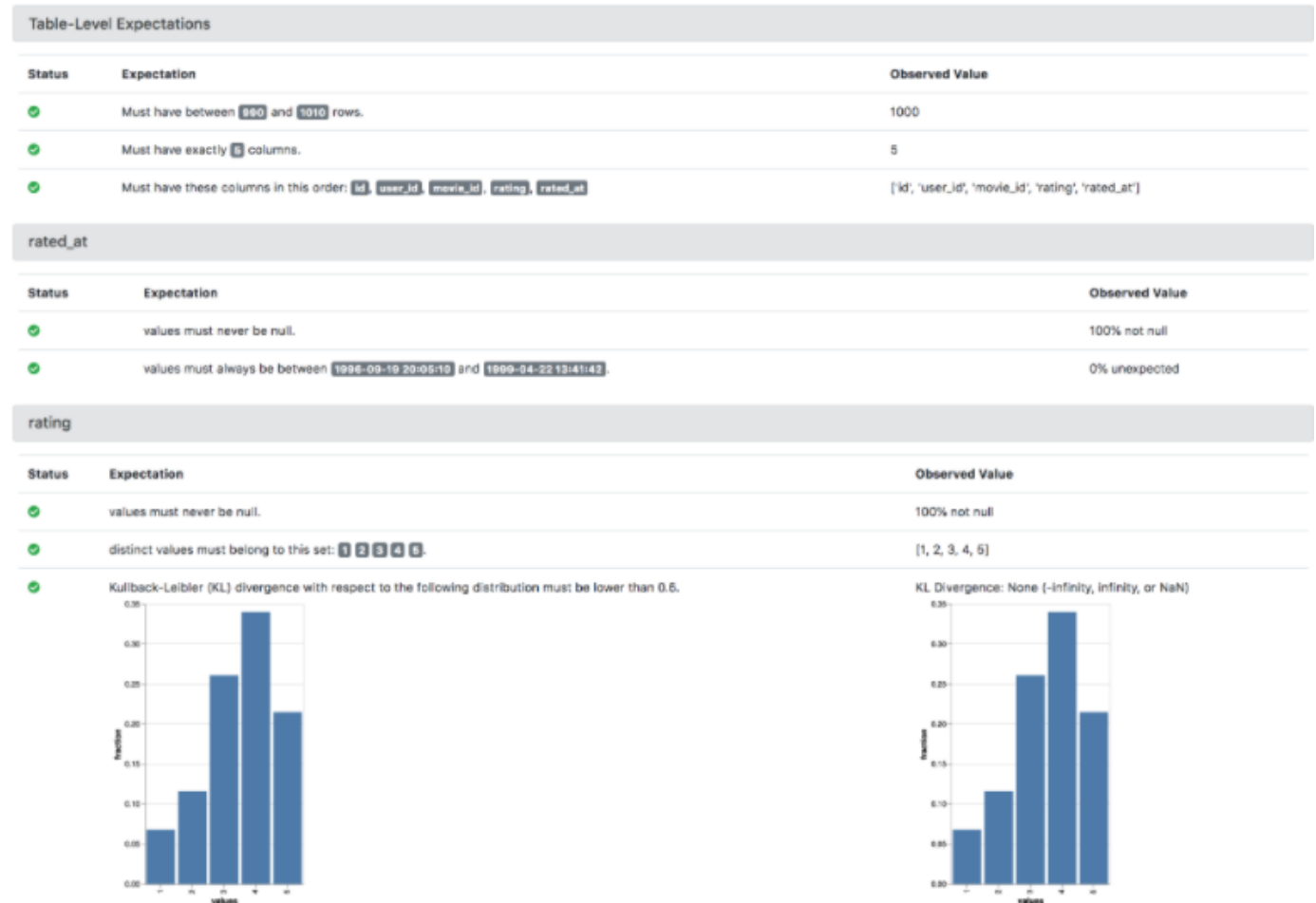


GREAT EXPECTATIONS



GE

- Сохранение и документирование статистики
- Test | CI процесс



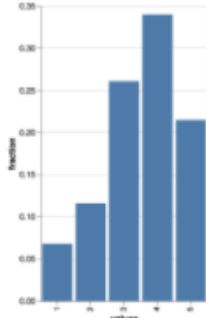
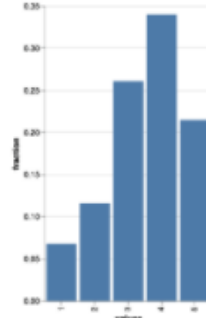
<https://greatexpectations.io/>

GE

- Документированные
- Тестируемые
- Стабильные


Table-Level Expectations		
Status	Expectation	Observed Value
✓	Must have between 900 and 1010 rows.	1000
✓	Must have exactly 5 columns.	5
✓	Must have these columns in this order: id, user_id, movie_id, rating, rated_at	['id', 'user_id', 'movie_id', 'rating', 'rated_at']

rated_at		
Status	Expectation	Observed Value
✓	values must never be null.	100% not null
✓	values must always be between 1996-09-19 20:05:19 and 1999-04-22 13:41:42.	0% unexpected

rating		
Status	Expectation	Observed Value
✓	values must never be null.	100% not null
✓	distinct values must belong to this set: 1 2 3 4 5.	[1, 2, 3, 4, 5]
✓	Kulback-Leibler (KL) divergence with respect to the following distribution must be lower than 0.6. 	KL Divergence: None (-infinity, infinity, or NaN) 

<https://greatexpectations.io/>

GE ACTION


 great_expectations [Home](#) / [Validations](#) / [locations.demo](#) / 2020-09-16T22:46:07.428761+00:00

Actions

Validation Filter:

Show All

Failed Only

 How to Edit This Suite

Show Walkthrough

Table of Contents

[Overview](#)

[Table-Level Expectations](#)

[dropoff_location](#)

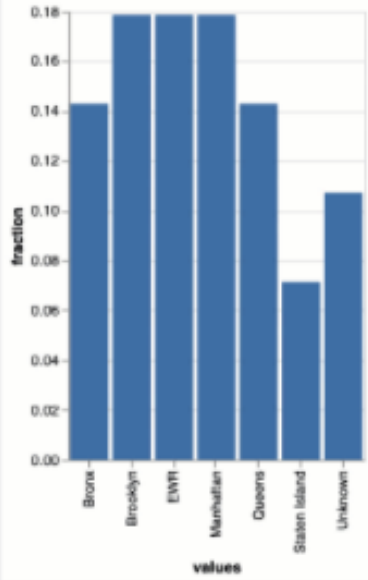
[num_rides](#)

✖

5 unexpected values found. ≈10.64% of 47 total rows.

✖

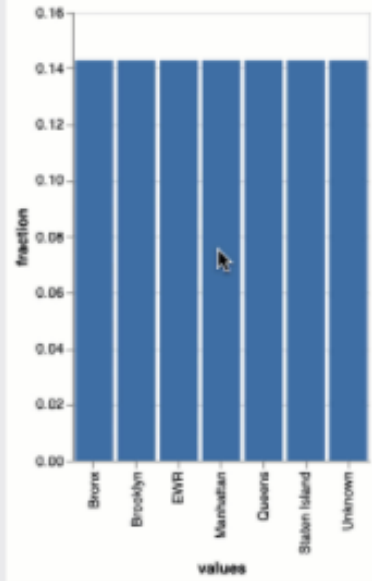
Kullback-Leibler (KL) divergence with respect to the following distribution must be lower than 0.01.



values	fraction
Bronx	0.145
Brooklyn	0.18
EWR	0.18
Manhattan	0.18
Queens	0.145
Staten Island	0.075
Unknown	0.11

≈89.362% not null

KL Divergence: ≈0.04449

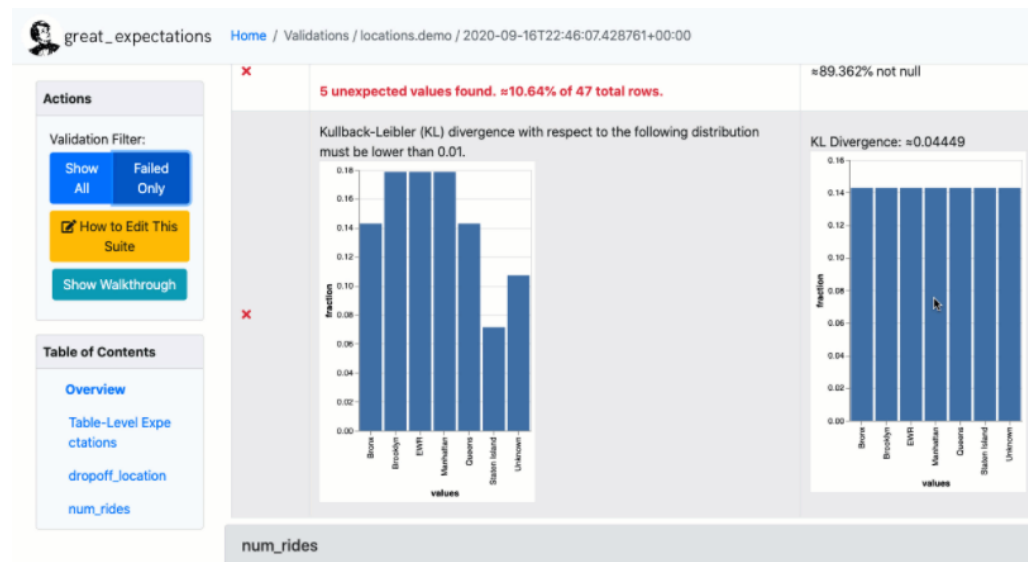


values	fraction
Bronx	0.145
Brooklyn	0.145
EWR	0.145
Manhattan	0.145
Queens	0.145
Staten Island	0.145
Unknown	0.145

num_rides

GE ACTION

https://github.com/great-expectations/great_expectations_action



GE JUPYTER NOTEBOOK

FEATURE PROBLEM



FEATURES



FEATURES



user_id	dt	feature_1	feature_n
777	2020-12-07 00:00:00	88.77	29.01

FEATURES



user_id	dt	feature_1	feature_n
777	2020-12-07 00:00:00	88.77	29.01



- SQL / Spark
- Python
- Scala

FEATURES



user_id	dt	feature_1	feature_n
777	2020-12-07 00:00:00	88.77	29.01



FEATURES



user_id	dt	feature_1	feature_n
777	2020-12-07 00:00:00	88.77	29.01

user_id	promo_id	dt	feature_1	feature_n
777	AFP99BC	2020-12-07 00:00:00	1	shop



FEATURES



user_id	dt	feature_1	feature_n
777	2020-12-07 00:00:00	88.77	29.01

- ↓
- SQL / Spark
 - Python
 - Scala
- ↑

user_id	promo_id	dt	feature_1	feature_n
777	AFP99BC	2020-12-07 00:00:00	1	shop



FEATURES



user_id	dt	feature_1	feature_n
777	2020-12-07 00:00:00	88.77	29.01

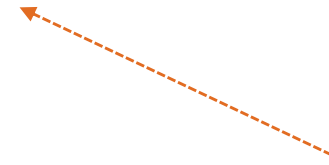


user_id	promo_id	dt	feature_1	feature_n
777	AFP99BC	2020-12-07 00:00:00	1	shop

FEATURES



user_id	dt	feature_1	feature_n
777	2020-12-07 00:00:00	88.77	29.01



user_id	promo_id	dt	feature_1	feature_n
777	AFP99BC	2020-12-07 00:00:00	1	shop

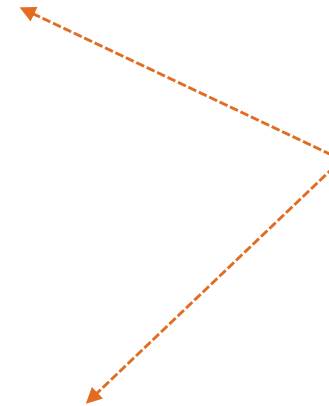
FEATURES



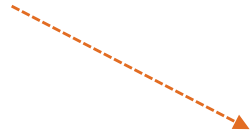
user_id	dt	feature_1	feature_n
777	2020-12-07 00:00:00	88.77	29.01



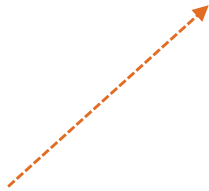
user_id	promo_id	dt	feature_1	feature_n
777	AFP99BC	2020-12-07 00:00:00	1	shop



FEATURES



- **SQL / Spark**
- **Python**
- **Scala**



FEATURES

user_id	promo_id	dt	feature_ds1_1	feature_ds1_n	feature_ds2_1	feature_ds2_n
777	AFP99BC	2020-12-07	88.77	29.01	1	shop



! FEATURES



- **Меньше времени на пайплайны**
(мы не DE)



- **Меньше повторений чужой работы**
- **Данные для исследований != прод**



- **Согласованность данных**
(consistency)

?

- Хранить и делиться данными (в real-time)
- Поставлять данные в большом кол-во инструментов
(готов работать со всеми ML инструментами)
- Ускорить работу
(устранить DE задачи)
- Сделать БД, которая не будет восприниматься, как БД

FEATURE STORE

- Michelangelo (Uber): **InfoQ Talk 2019**
- Hopsworks (Logical Clocks): **Bay Area AI Talk 2019**
- Feast (GoJek): **HasGeek TV Talk 2019**
- Zipline (AirBnB): **Spark/AI Summit 2019**
- Metaflow (Netflix): **MLOps 2019**
- FBLearner (Facebook): **Twiml Interview 2018**
- Feature Factory (Databricks): **Spark/AI Summit 2019**
- Online Feature Store (Zomato): **HasGeek TV Talk 2019**
- Galaxy (Pinterest): **Slides from 2019**
- SurveyMonkey Feature Store: **Slides from 2020**
- Comcast Feature Store: **Spark Summit 2018**
- Wix Feature Store: **Wix Engineering Talk 2019**
- Twitter Feature Store: **MLOps 2019**


FEATURE STORE

- Michelangelo (Uber): **InfoQ Talk 2019**
- Hopsworks (Logical Clocks): **Bay Area AI Talk 2019**
- Feast (GoJek): **HasGeek TV Talk 2019**
- Zipline (AirBnB): **Spark/AI Summit 2019**
- Metaflow (Netflix): **MLOps 2019**
- FBLearner (Facebook): **Twiml Interview 2018**
- Feature Factory (Databricks): **Spark/AI Summit 2019**
- Online Feature Store (Zomato): **HasGeek TV Talk 2019**
- Galaxy (Pinterest): **Slides from 2019**
- SurveyMonkey Feature Store: **Slides from 2020**
- Comcast Feature Store: **Spark Summit 2018**
- Wix Feature Store: **Wix Engineering Talk 2019**
- Twitter Feature Store: **MLOps 2019**

FEATURE STORE

```
# Создать драйвер и зарегистрировать фичи
fs = FeatureSet("all_users_agg_m")
fs.infer_fields_from_df(df)
client.apply(fs)

# загрузить данные в FeatureStore
client.ingest(fs, df)
```



The diagram illustrates the process of creating a FeatureSet from a DataFrame and then ingesting it into a FeatureStore. A dashed orange arrow points from the `df` parameter in the `client.ingest(fs, df)` line of the code to the DataFrame table. Another dashed orange arrow points from the `fs` parameter in the same line to the FeatureSet configuration block.


user_id	dt	feature_1	feature_n
777	2020-12-07 00:00:00	88.77	29.01

```
name: all_users_aggr_m
entities:
  - name: user_id
  - valueType: INT32
features:
  - name: dt
  - valueType: TIMESTAMP
  ...
```

FEATURE STORE

```
# Создать драйвер и зарегистрировать фичи
fs = FeatureSet("all_users_agg_m")
fs.infer_fields_from_df(df)
client.apply(fs)

# загрузить данные в FeatureStore
client.ingest(fs, df)
```



The diagram consists of two dashed orange arrows. The first arrow originates from the 'user_id' column of the table above and points to the 'fs' variable in the code block on the left. The second arrow originates from the 'feature_1' and 'feature_n' columns of the table and points to the 'Entity' and 'Feature' lists below.

user_id	dt	feature_1	feature_n
777	2020-12-07 00:00:00	88.77	29.01

•Entity

- Organization ID Type
- User ID
- №
- Hash

•Feature

- Все остальное

FEATURE STORE

```
# выбрать колонки
featute_list = {
    "all_users_aggr_m: user_id",
    "all_users_aggr_m: feature_1"
}

# загрузить из драйвера
features = client.get_batch_feature(
    entity_rows = drivers,
    feature_ids = featute_list
).to_dataframe()
```



user_id	feature_1
777	88.77

FEATURE STORE

```
# выбрать колонки
featute_list = {
    "all_users_aggr_m: user id",
    "all_users_aggr_m: feature_1"
}

# загрузить из драйвера
features = client.get_batch_feature(
    entity_rows = drivers,
    feature_ids = featute_list
).to_dataframe()
```


FEATURE STORE

```
# выбрать колонки
featute_list = {
    "all_users_aggr_m: user id",
    "all_users_aggr_m: feature_1"
}

# загрузить из драйвера
features = client.get_batch_feature(
    entity_rows = drivers,
    feature_ids = featute_list
).to_dataframe()
```

```
name: all_users_aggr_m
entities:
  - name: user_id
  - valueType: INT32
features:
  - name: dt
  - valueType: TIMESTAMP
  ...
```

FEATURE STORE

```
# выбрать колонки
featute_list = {
    "all_users_aggr_m: feature_1"
}

# загрузить из драйвера
features = client.get_batch_feature(
    entity_rows = drivers,
    feature_ids = featute_list
).to_dataframe()
```



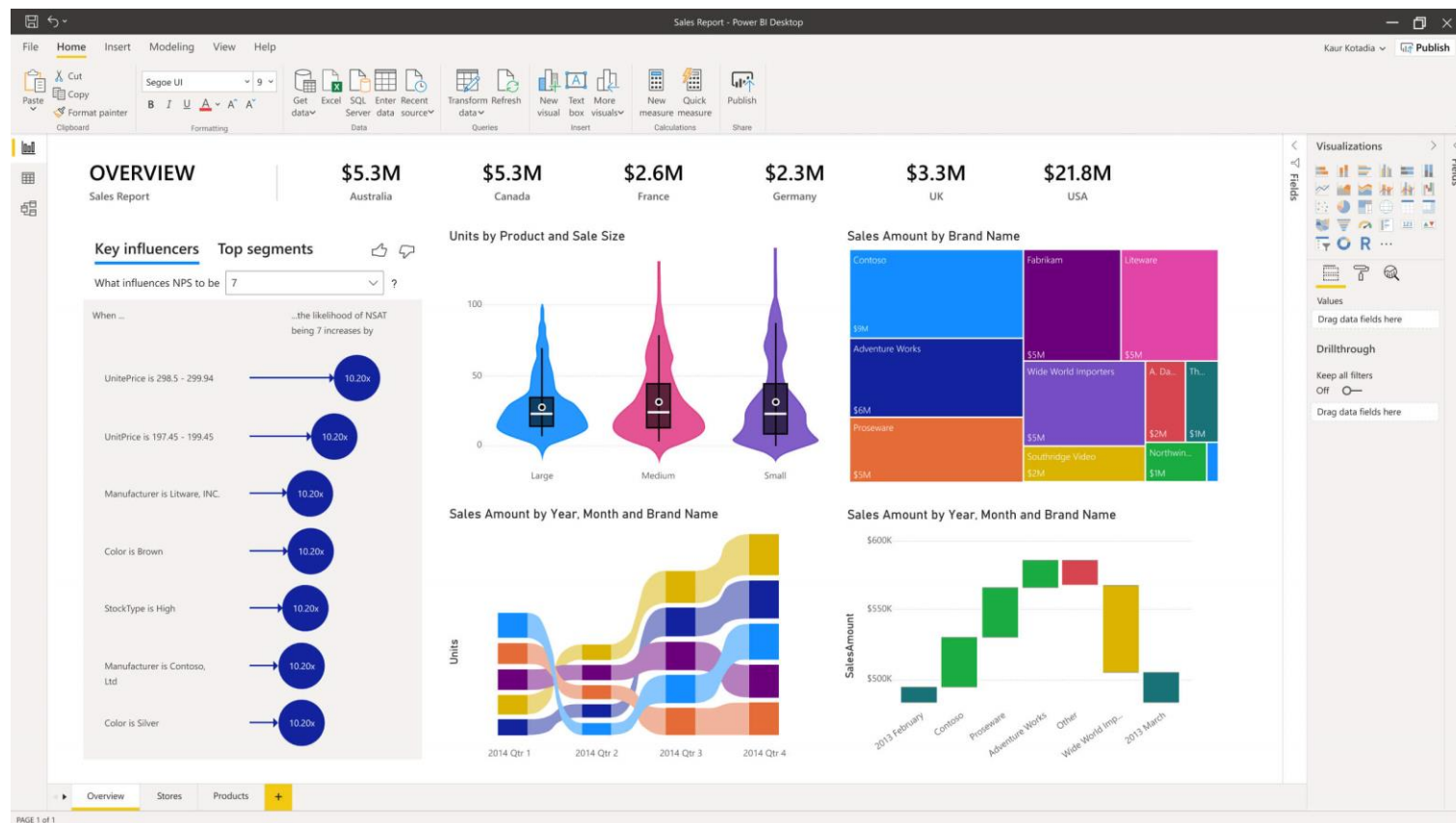
user_id	feature_1
777	88.77

FS != DWH



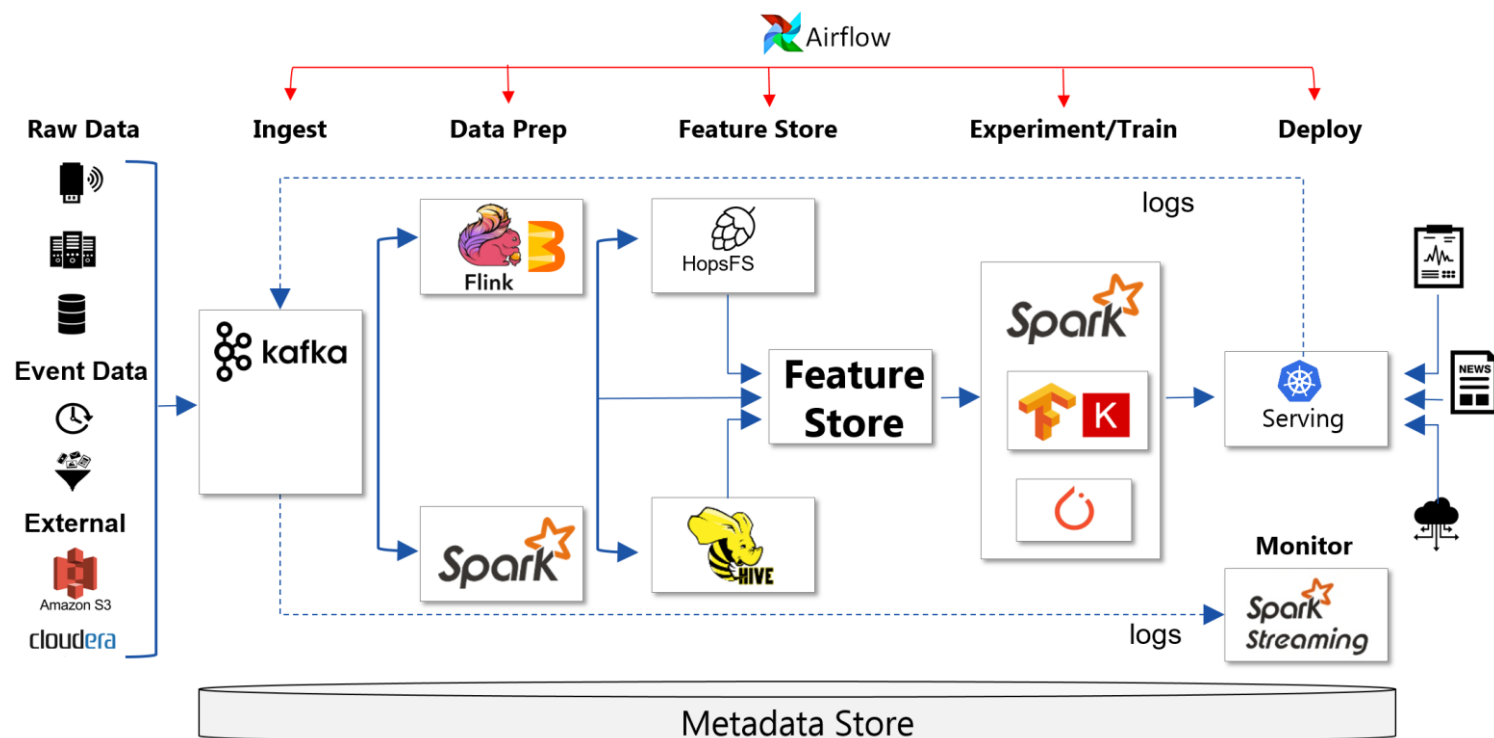
FS != DWH

- **DS данные – это инсайты поведения, а не статистика для BI**



FS != DWH

- DS данные – это инсайты поведения, а не статистика для BI
- Offline / online разделение в базе



FS != DWH

- DS данные – это инсайты поведения, а не статистика для BI
- Offline / online разделение в базе
- Включены валидаторы данных ([Great Expectations](#) / [Deequ](#))

FS != DWH

- DS данные – это инсайты поведения, а не статистика для BI
- Offline / online разделение в базе
- Включены валидаторы данных ([Great Expectations](#) / [Deequ](#))
- Фиксация времени и изменений в данных

FS != DWH

Параметр	DWH	FS
Данные	Реляционные схемы	Типизированные данные, зависимость от модели
МетаДанные	Стандартное мета-хранилище для поиска и обслуживания	DWH + статистики и дрифты
Пользователи	BI	DS
Порядок доступа	Скрипт запросы	Фильтры, цикличное обращение, ключ к фичам
Валидация данных	-	GE (python) / D (Scala)
API	SQL	Python, Scala, Java

FS != DWH

Параметр	DWH	FS
Данные	Реляционные схемы	Типизированные данные, зависимость от модели
МетаДанные	Стандартное мета-хранилище для поиска и обслуживания	DWH + статистики и дрифты
Пользователи	BI	DS
Порядок доступа	Скрипт запросы	Фильтры, цикличное обращение, ключ к фичам
Валидация данных	-	GE (python) / D (Scala)
API	SQL	Python, Scala, Java

СДЕЛАЕМ ВЫБОР



FEATURE STORE

Platform	Open Source	Offline	Online	Metadata
Hopsworks	AGPL-V3	Hudi/Hive	MySQL Cluster	DB Tables, Elasticsearch
Michelangelo	n/a	Hive	Cassandra	KV Entries
Feast	Apache V2	BigQuery	BigTable/Redis	DB Tables

FEATURE STORE

Platform	Open Source	Offline	Online	Metadata
Hopsworks	AGPL-V3	Hudi/Hive	MySQL Cluster	DB Tables, Elasticsearch
Michelangelo	n/a	Hive	Cassandra	KV Entries
Feast	Apache V2	BigQuery	BigTable/Redis	DB Tables

FEATURE STORE

	Platform	Open Source	Offline	Online	Metadata
https://github.com/logicalclocks/hopsworks	Hopsworks	AGPL-V3	Hudi/Hive	MySQL Cluster	DB Tables, Elasticsearch
	Michelangelo	n/a	Hive	Cassandra	KV Entries
https://github.com/feast-dev/feast	Feast	Apache V2	BigQuery	BigTable/Redis	DB Tables

FEATURE STORE

<https://github.com/logicalclocks/hopsworks>

Platform	Open Source	Offline	Online	Metadata
Hopsworks	AGPL-V3	Hudi/Hive	MySQL Cluster	DB Tables, Elasticsearch
Michelangelo	n/a	Hive	Cassandra	KV Entries
Feast	Apache V2	BigQuery	BigTable/Redis	DB Tables

<https://github.com/feast-dev/feast>



<https://www.youtube.com/watch?v=UNailXoilrY>



<https://github.com/ComcastSamples/OSCON-2019-End-to-end-ML-feature-streaming-with-Kubeflow-Kafka-and-Redis-demo-code>

КАК РАБОТАЕТ?



ЗАДАНИЕ



ЗАВЕРШИТЬ ЗАДАНИЯ 1, 2, 3

- Сделайте выбор между Feast / HopsWorks