data

#SBERTEAM

Data Quality, or How to Test Recommendations

HeisenBug 2024

апрель ‘24

О докладчике

- MLOps developer в Сбере.
- Разрабатываю SDK и технологически развиваю платформу.
- Есть опыт в развитии enterprise продуктов и мануальном тестировании.

О продукте

Рекомендательная платформа Сбера.

Наша задача создавать рекомендательные системы, которые понимают пользователей и точно попадают в их запросы.

Цель доклада

- Рассказать всем о Data Quality и опыте внедрения.
- Донести важность подхода, показать на примере как тестировать данные.
- Поделиться опытом реализации

Рекомендательная платформа Сбера

Data user

Back-end

DS SDK

ML Tools

Models, Services

User

Интеграционные сервисы платформы

Go\Py

Front-end

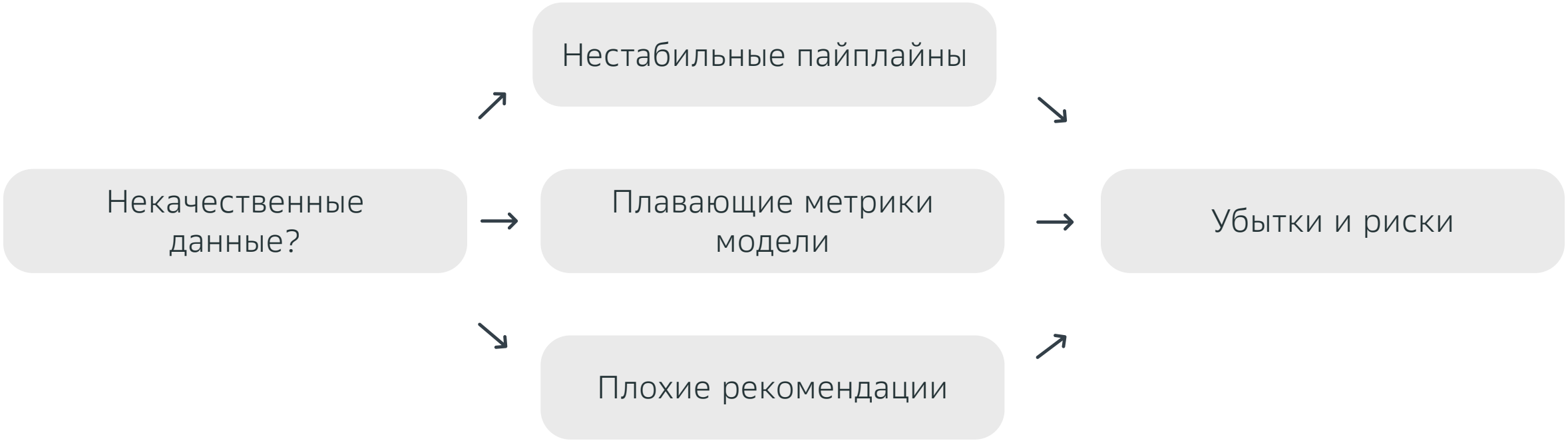
RecSys Studio Web \ Airflow

Channels

Client

Что такое Data Quality?

Проблема


Некачественные данные?
Нестабильные пайплайны
Плавающие метрики модели
Плохие рекомендации
Убытки и риски

Цели и задачи

- Выработка управляемого подхода к обеспечению соответствия данных нуждам их потребителей.
- Определение стандартов и спецификаций механизмов контроля качества данных как составной части жизненного цикла данных.
- Определение и внедрение процессов измерения, мониторинга и учета уровня качества данных.
- Выявление и поддержка использования возможностей по повышению качества данных посредством внесения изменений в системы и процессы, а также осуществление деятельности по
  проведению измеримых улучшений качества данных на основе требований их потребителей.

Что такое качественные данные?

Степень пригодности данных к решению задачи формирования рекомендаций и/или иных дата продуктов, в соответствии с определенными бизнес правилами.

Виды проверок

- Спецификация
- Логика
- Статистики

Базовый

Сумма покупки должна быть выше 0.

Значения параметра «пол» может быть только в множестве -> (М, Ж)

При наличие даты платежа, должно быть значение суммы платежа.

Задержка попадания события в kafka < 10с

Регулярные

Продвинутый

Что такое качественные данные?

Степень пригодности данных к решению задачи формирования рекомендаций и/или иных дата продуктов, в соответствии с определенными бизнес правилами.

Виды проверок

- Спецификация
- Логика
- Статистики

Базовый

Сумма покупки должна быть выше 0.

Значения параметра «пол» может быть только в множестве -> (М, Ж)

При наличие даты платежа, должно быть значение суммы платежа.

Задержка попадания события в kafka < 10с

Регулярные

У 80% «горячих» пользователей, должна быть рекомендация.

К каждому пользователю, выдается более 100 рекомендаций.

Значение в столбце содержит валидное значение для geojson

Продвинутые

Что такое качественные данные?

Степень пригодности данных к решению задачи формирования рекомендаций и/или иных дата продуктов, в соответствии с определенными бизнес правилами.

Виды проверок

- Спецификация
- Логика
- Статистики

Базовый

Сумма покупки должна быть выше 0.

Значения параметра «пол» может быть только в множестве -> (М, Ж)

При наличие даты платежа, должно быть значение суммы платежа.

Задержка попадания события в kafka < 10с

Регулярные

У 80% «горячих» пользователей, должна быть рекомендация.

К каждому пользователю, выдается более 100 рекомендаций.

Значение в столбце содержит валидное значение для geojson

Продвинутые

Оценка косинусного расстояния полученных эмбеддингов, перед обучением моделей.

Проверка эмбеддингов перед обучением модели на классических алгоритмах ML, оценка полученных метрик.

Качество данных как часть архитектуры

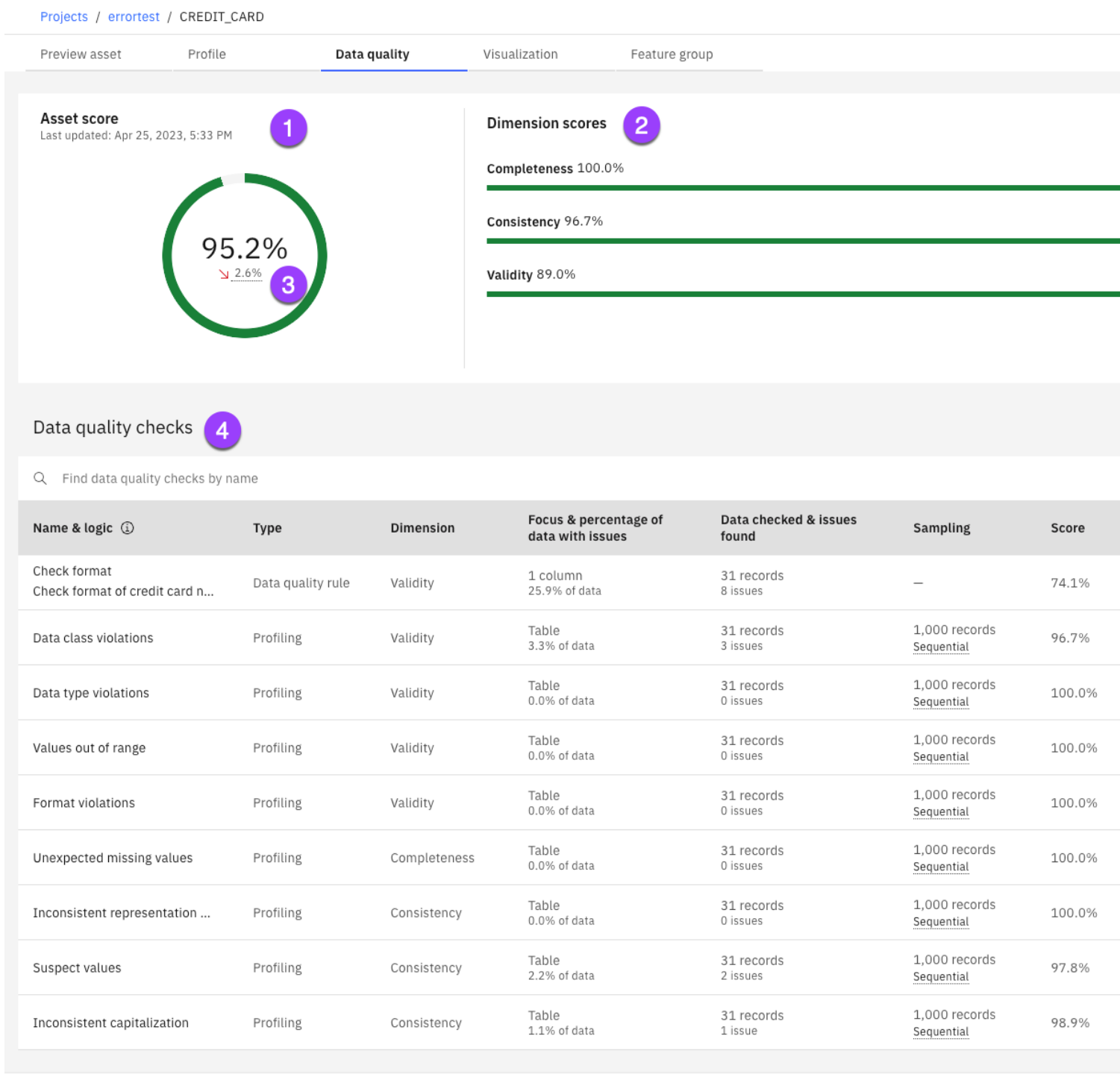
Индустриальный опыт

Cloud решения

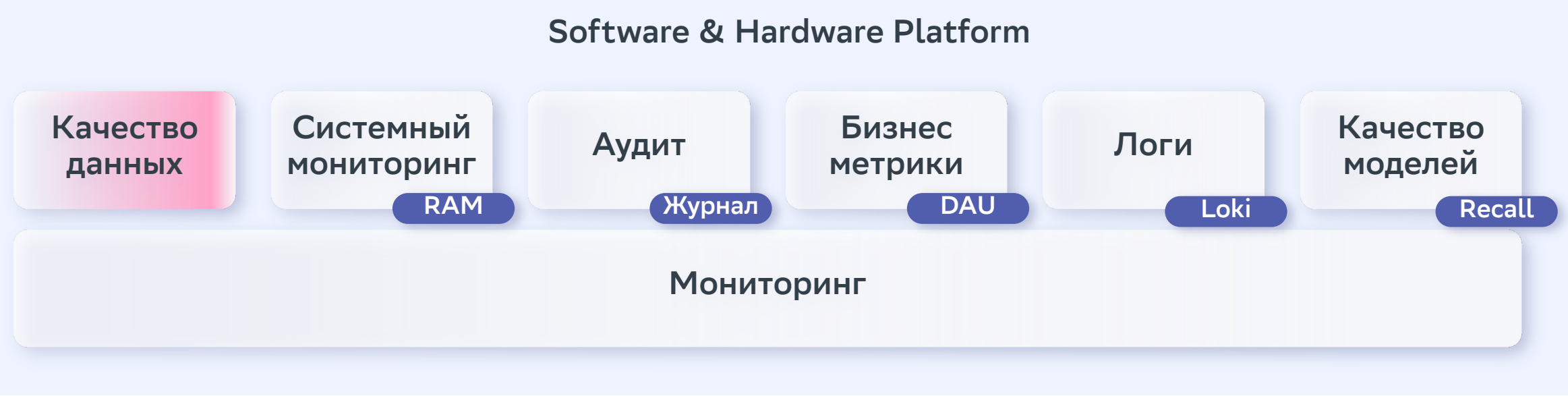
Data Quality - единый UI для проверок, вместе или отдельно с/от storage/compute инструментами

Представлены как:

- Компонент единого Cloud решения (Informatica, IBM, Google),
- Входной консоли дата платформы (Cloudera)
- Отдельного компонента дата каталога (OMD, DataHub),
- Отдельного механизма (SDP DQ, Talend, GX, Soda, Whylogs)



Часть общего мониторинга



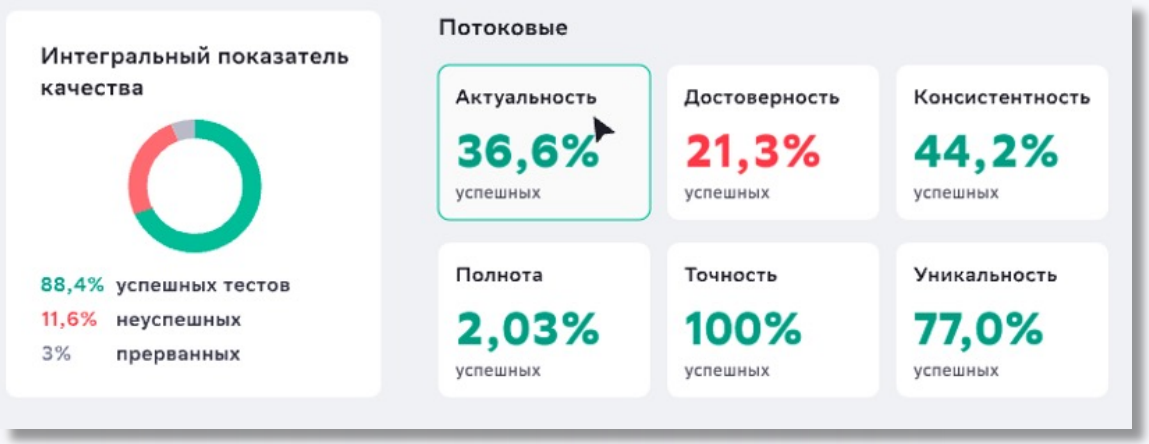
Анализ, оценка, метрики

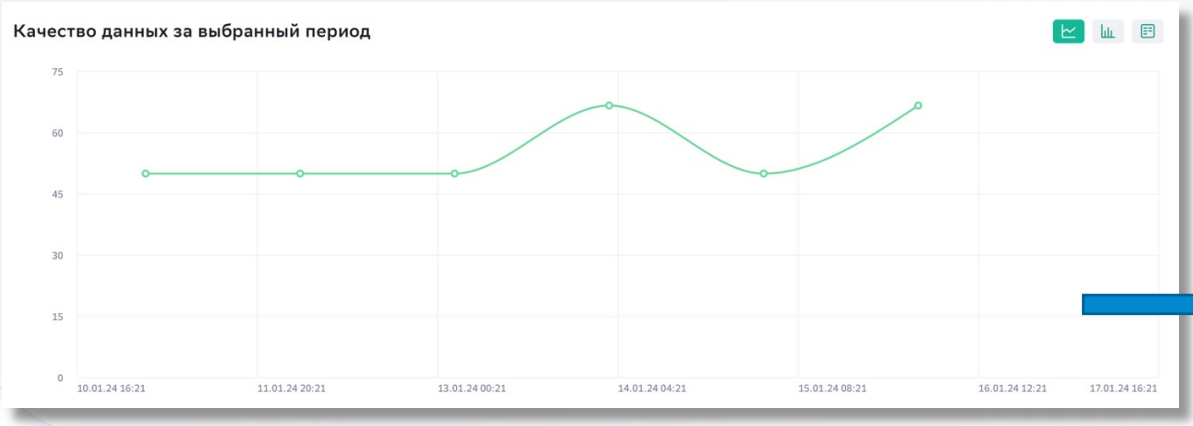
Сценарии

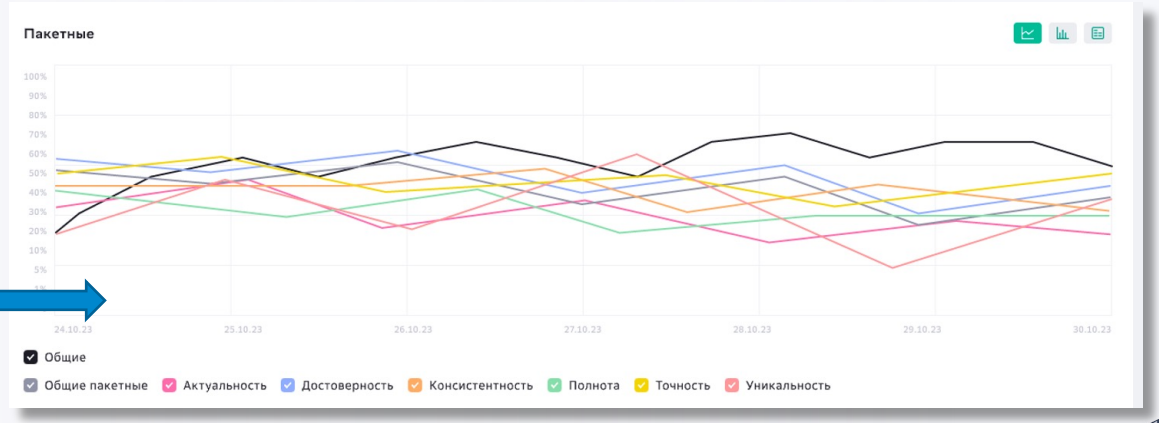
- Отчёт за месяц по интересующим показателям
- Отчёт за день по интегральному показателю
- Повышение покрытия отдельной метрики

Для критичных проверок

- Откат версий модели
- Выдача дефолтных рекомендаций

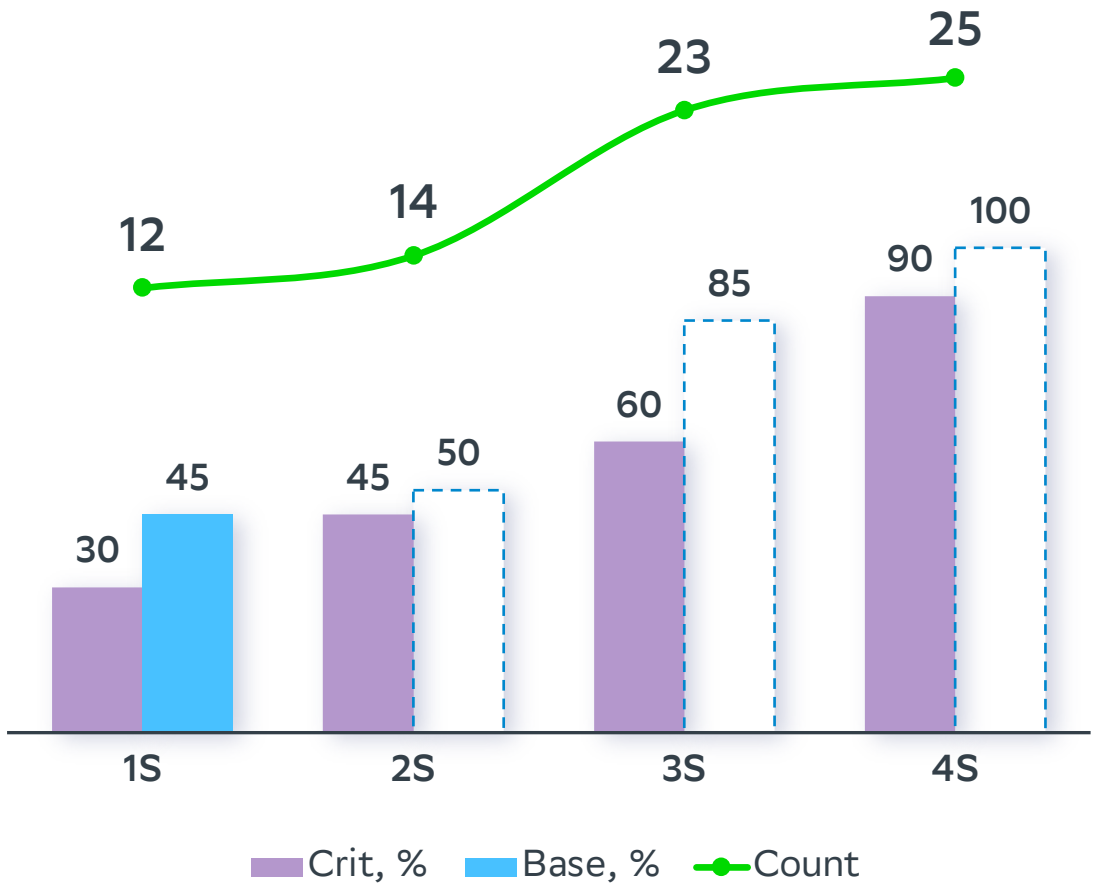






Как внедрять?

Начать с простых проверок и брать количество



План

- Начать со спецификаций
- Соблюдение простой логики
- Соблюдение бизнес логики
- Расхождения статистик
- Оценка потоковых данных
- Адаптировать со временем

всего **30 шт**

Успешных **80 % +**

Следующий этап:

Внедрять сложные проверки

Инструменты тестирования

Выбор фреймворка

	Great Ex	Deequ	Soda Core
Функциональность	Богатая	Достаточная	Ограниченная
Comment	Тяжелый инструмент	Слабое Python API	Многих фичей нет в Open Source
Кол-во проверок	300+	~50	25+
SDK / API	Перегружен	Удобный	Удобный
Источники	Pandas, Spark, SQL, *	Spark	Pandas, Spark, SQL
Активность	9,4 K \ High	3,1K \ Low	1,7 \ Normal
Интеграции	Много	Spark	Мало
Документация	Очень тяжелая	Простые example	Достаточная

Достойны упоминания

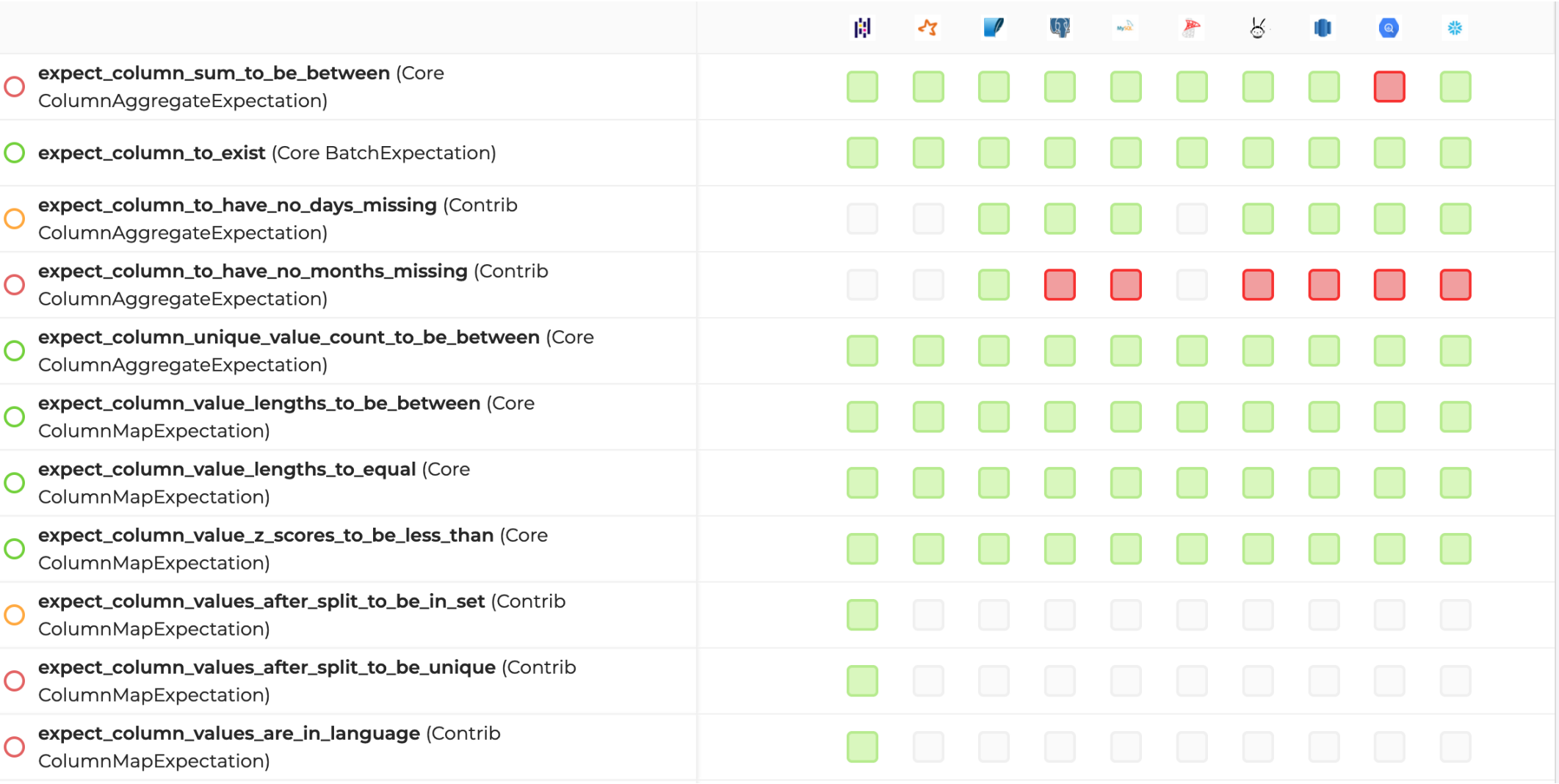
- Apache Griffin
- Datafold
- Data build tool
- Pandera

Пример проверки с Great Expectations

```python
import great_expectations as gx
from pyspark.sql import SparkSession

#example spark source
spark = SparkSession.builder.appName("Demo suite for GX").getOrCreate()
dataframe = spark.read.table("yellow_tripdata_sample")

#set up
context = gx.get_context()
# optionals fields
datasource = context.sources.add_spark("example_source")
data_asset = datasource.add_dataframe_asset(name="example_asset")
my_batch_request = data_asset.build_batch_request(dataframe=dataframe)
expectation_suite_name = "Example suite"
context.add_or_update_expectation_suite(expectation_suite_name=expectation_suite_name)

#connect to data
validator = context.get_validator(
    batch_request=my_batch_request,
    expectation_suite_name=expectation_suite_name,
)
#create expectations
validator.expect_column_values_to_not_be_null("pickup_datetime", meta={"level": "error"})
validator.save_expectation_suite(discard_failed_expectations=False)

#validate data
checkpoint = context.add_or_update_checkpoint(
    name="my_quickstart_checkpointee23",
    validator=validator,
    expectation_suite_name=expectation_suite_name,
    runtime_configuration={"result_format": {"result_format": "COMPLETE"}},  # to return counts of unexpected values
)
checkpoint_result = checkpoint.run()

#view result
context.view_validation_result(checkpoint_result)
```

- Создать \ получить Datacontext
- Подключиться к источнику (память, база, файловая система)
- Создать «ожидания» (проверки)
- Запустить проверку
- Посмотреть \ сохранить результат

Много стандартных проверок



Пример с использованием Deequ

```
check_deq = Check(spark, CheckLevel.Warning, "Demo Check")

checkResult = (
    VerificationSuite(spark)
    .onData(df)
    .addCheck(
        check_deq.hasSize(lambda x: x ⩾ 3)
        .hasMin("b", lambda x: x == 0)
        .isComplete("c")
        .isUnique("a")
        .isContainedIn("a", ["foo", "bar", "baz"])
        .isNonNegative("b")
    )
    .run()
)
```

- Не сильно перегруженный API
- Меньше проверок
- Слабая поддержка Python
- Хорошо умеет в Big Data
- Для начала подойдет лучше

Базовый пример профилирования WhyLogs

- Получить «log» источника
- Построить профиль
- Наложить ограничения на профиль
- Запустить проверки и получить отчет

```
import whylogs as why

# Log the data frame. This equivalent to why.log(retail_daily) and why.log(data=retail_daily)
results = why.log(pandas=retail_daily)

# Get the Results
profile = results.profile()

# Head down to Display a Log for explination
profile.view().to_pandas()
```

column	counts/n	counts/null	types/integral	types/fractional	types/boolean	types/string	types/object	cardinality/est	cardinality/upper_1	cardinality/lower_1	...
Purchase Canceled	909	72	0	837	0	0	0	2.000000	2.000100	2.000000	...
Age at Transaction Date	909	0	0	909	0	0	0	25.000001	25.001250	25.000000	...
Transaction Week	909	0	909	0	0	0	0	1.000000	1.000050	1.000000	...
Store Type	909	0	0	0	0	909	0	4.000000	4.000200	4.000000	...
Product Category	909	0	0	0	0	909	0	6.000000	6.000300	6.000000	...
Gender	909	0	0	0	0	909	0	2.000000	2.000100	2.000000	...
Transaction ID	909	0	0	0	0	909	0	904.722898	916.565225	893.168643	...
Item Price	909	0	0	909	0	0	0	672.542875	681.346093	663.953801	...
Total Tax	909	0	0	909	0	0	0	800.975225	811.459552	790.745935	...
Product Category Code	909	0	909	0	0	0	0	6.000000	6.000300	6.000000	...
Transaction Day of Week	909	0	909	0	0	0	0	1.000000	1.000050	1.000000	...
City Code	909	1	0	908	0	0	0	10.000000	10.000500	10.000000	...
Transaction Batch	909	0	909	0	0	0	0	1.000000	1.000050	1.000000	...
Date of Birth	909	0	0	0	0	909	0	801.978113	812.475567	791.736016	...

Наша реализация

Сценарий #1. Добавления теста из коробки

Аудитория
- Пользователи продукта
  - Data Scientists
  - Data Engineers

Сценарии
- Добавление теста контроля характеристик качества

Алгоритм
- Зайти в UI, выбрать датасет, выбрать тест, run.

```
{
  "test_type": "valueInSet",
  "test_name": "valueInSet_ClickStream",
  "test_suite": "generic_checks",
  "quality_group": "accuracy",
  "priority": 1,
  "args": {
    "dataset": "db.clickstream",
    "partition": "*/date=${date}",
    "column_list": ["eventType"],
    "allowed_values": ["click", "like",
      "add_to_favorite", "order"]
  }
}
```

Конфигурирование тестов «из коробки» в Studio WEB

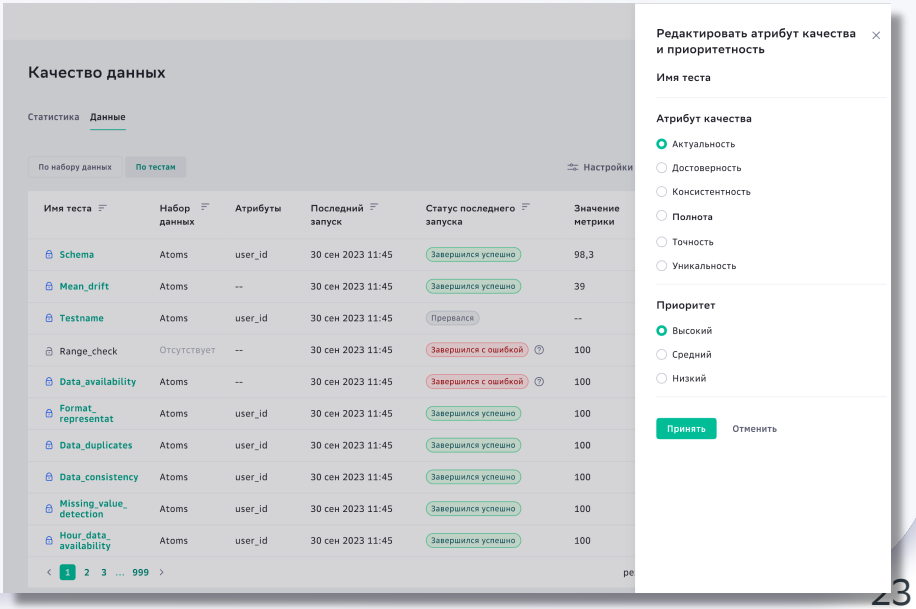
Сценарий #2. Добавления user теста

Аудитория
- Data Engineers
- Data Scientists

Сценарии
- Добавление пользовательских тестов (airflow)

Алгоритм
- Обновить DAG/task в Airflow
- Реализовать код чтения данных
- Добавить проверку
- Добавить код отправки результата в DQS
- Продуктизировать код
- Назначить показатель качества и приоритет

```python
import pydeequ
from pydeequ.checks import Check, CheckLevel
from pydeequ.verification import VerificationSuite

# Create and run test for dataframe df
check_deq = Check(spark, CheckLevel.Warning, "event_type_validness")
checkResult = (
    VerificationSuite(spark)
    .onData(df)
    .addCheck(
        check_deq.isContainedIn("eventType",
                                ["click", "like", "add_to_favorite", "order"])
    )
    .run()
)

# Send result to DQS from amazmedaglib.quality import DQSClient
dqs = DQSClient()
dqs.send_check(checkResult, group=ACCURACY, priority=3)
```



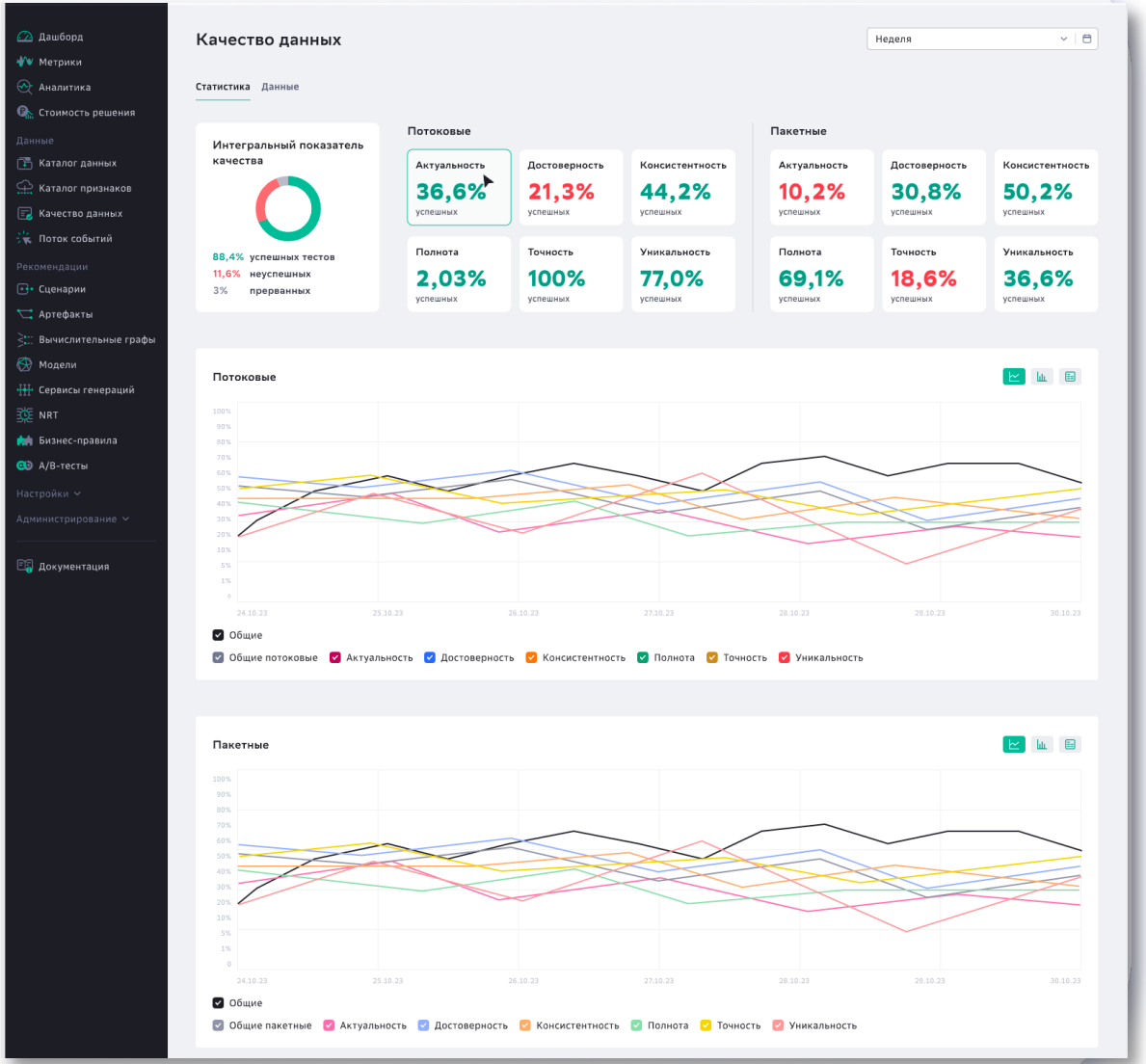
Сценарий #3. Просмотр статистики

Аудитория

- Старший менеджмент
- Владельцы продукта

Сценарии

- Принять решения на базе интегрального показателя качества
  - Общего
  - Для пакетных данных
  - Для потоковых данных



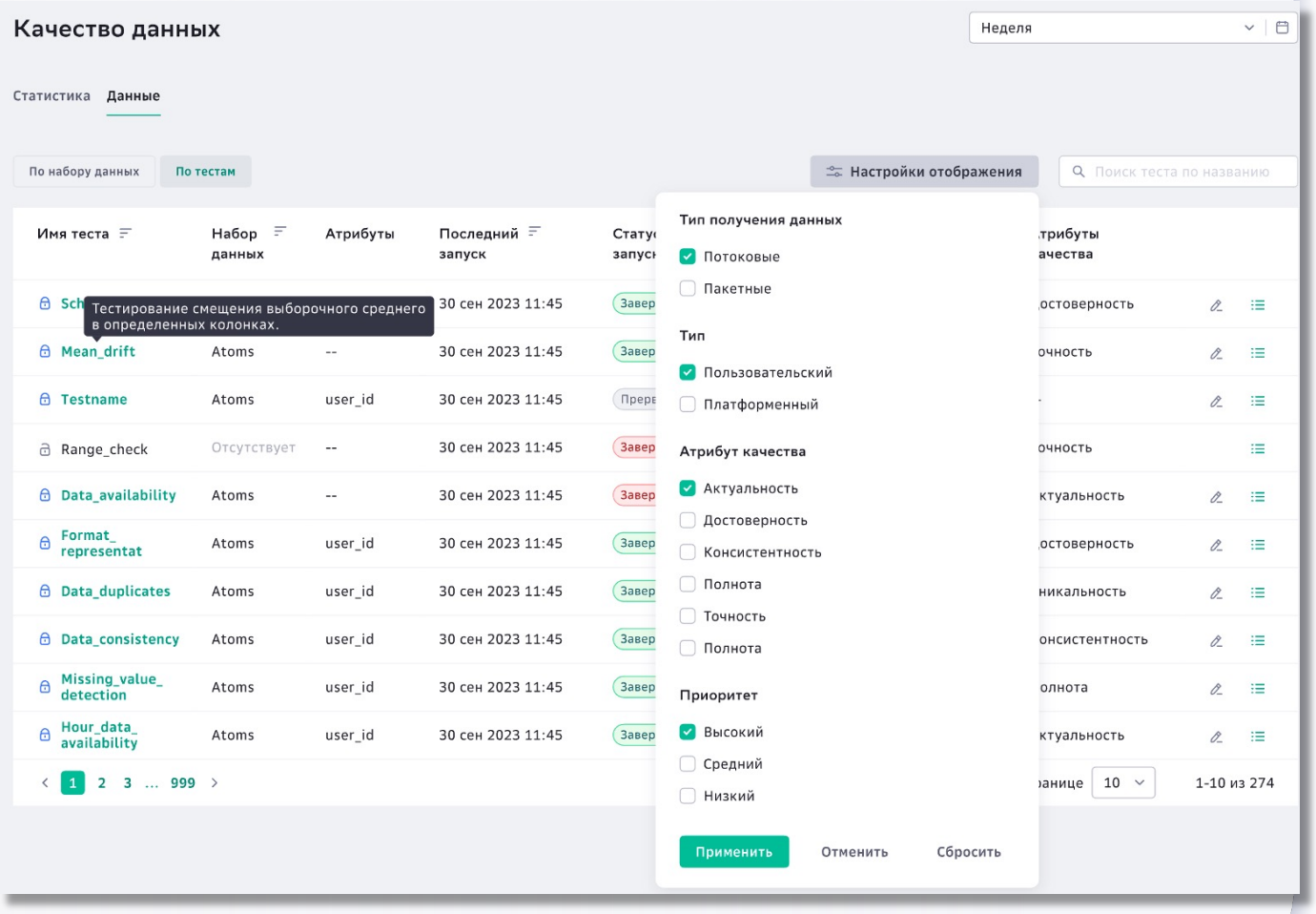
Сценарий #4. Просмотр деталей

Аудитория

- Лидеры направлений
- Data Engineers
- Data Scientists

Сценарии

- Проанализировать результаты отдельных тестов
- Проанализировать качество датасетов
- Проанализировать качество по группам
- Потоковые / пакетные
- Актуальность / достоверность
- Приоритет



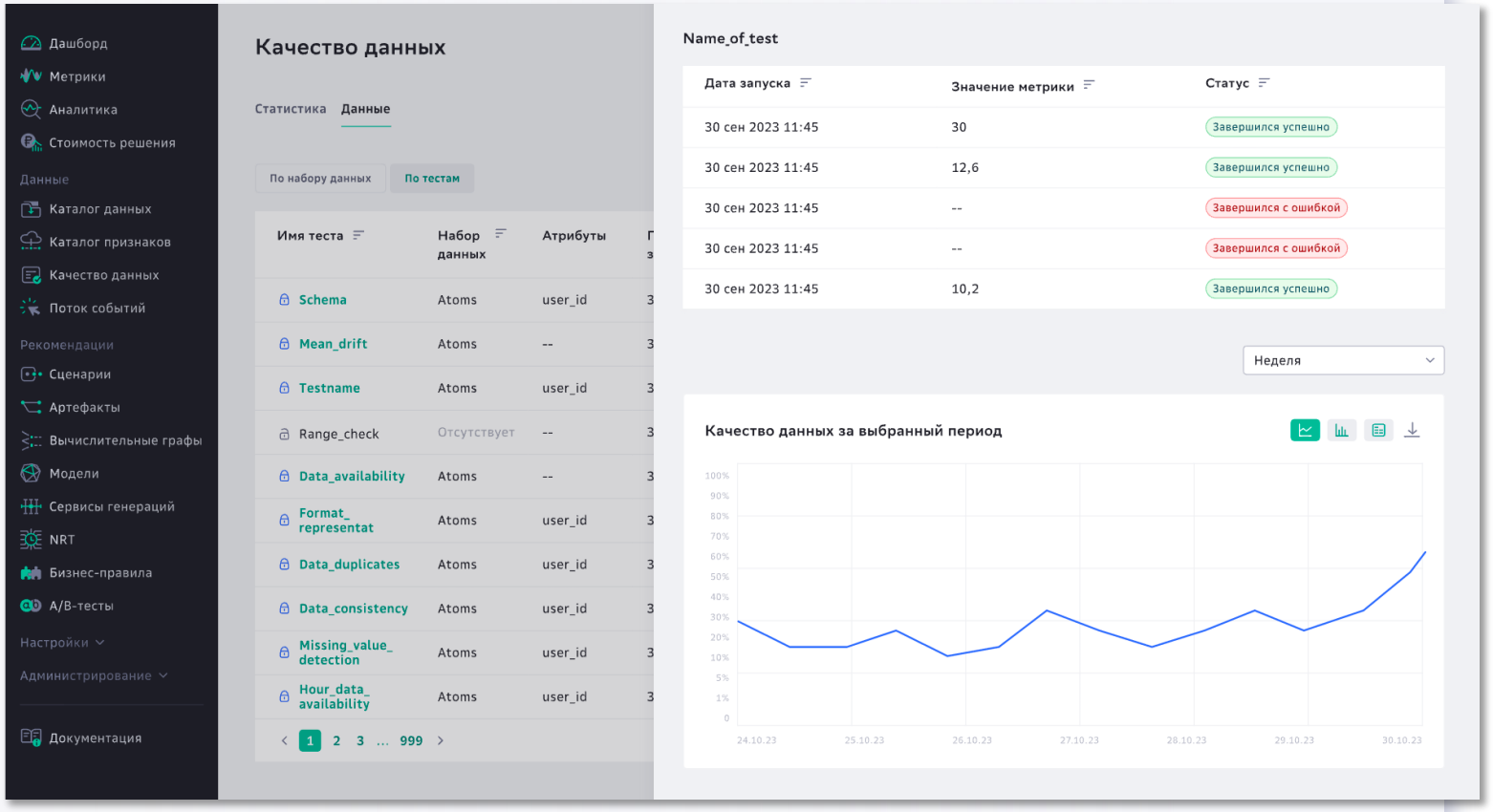
Сценарий #5. Анализ истории проверок

Аудитория

- Лидеры направлений
- Data Engineers
- Data Scientists

Сценарии

- Проанализировать историю качества данных теста на предмет его актуальности
- Ретроспектива



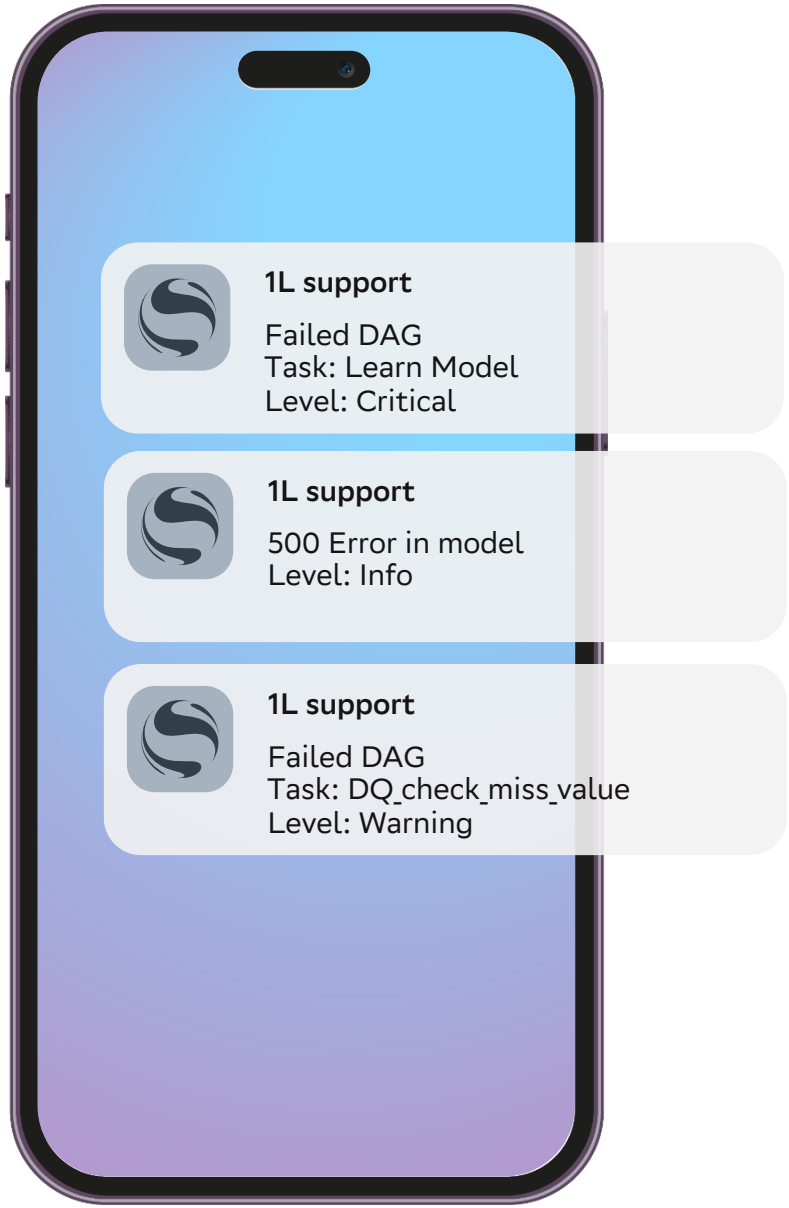
Алертинг

Результат каждого теста или тестовой группы, может по разному влиять на runtime. Какая-то группа проверок может не оказывать влияния на prom, но критически важно их считать.

Критичность*

- Инфо проверки
- Проверки без изменения статуса
- Критичные проверки, заканчивающие пайплайн

Prometheus



*Критичность каждого теста, зависит от сценария

Реализация на платформе

Выводы

Наши выводы

- Внедрять нужно снизу вверх
- Скорее всего у вас уже есть Data Quality
- Метрик «количество тестов» и «успешно», «неуспешно» достаточно.
- Это как и другие виды тестов, все понимают что нужно, но не делают.