От потоков данных до высококачественных ML-моделей

Секрет конструирования и тестирования признаков

SmartData
2023

Обо мне

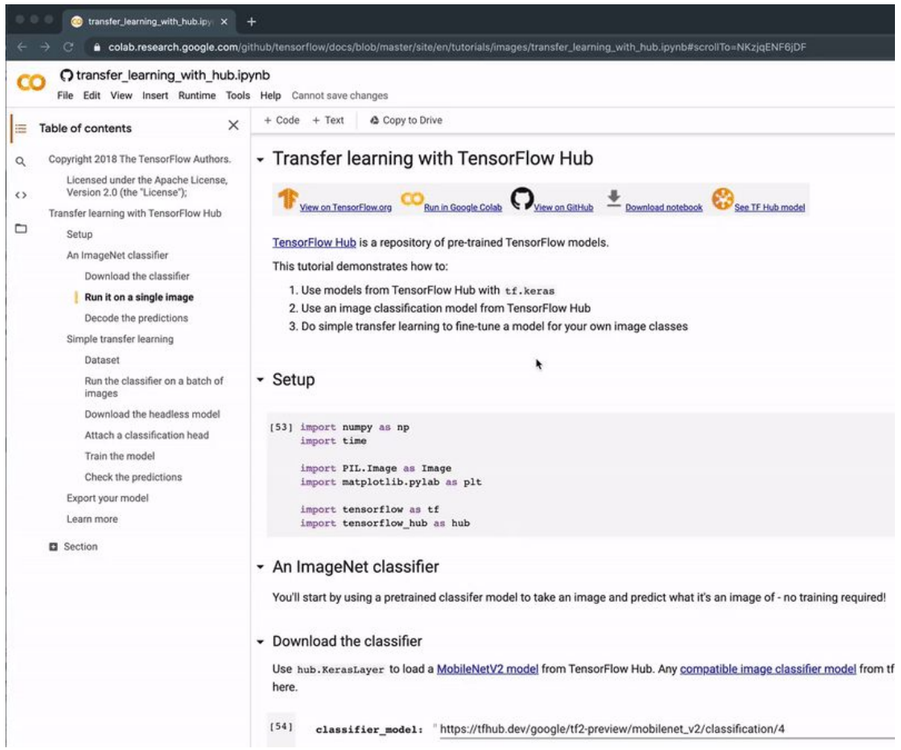
Юрий Гусев

linkedin.com/in/ygusev

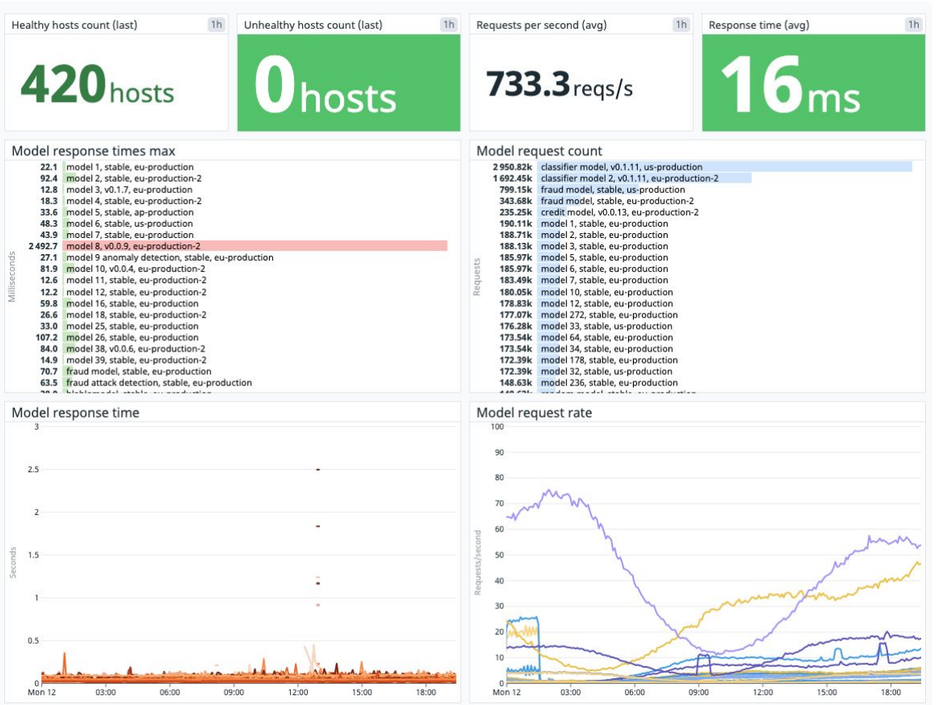
- Окончил МГТУ им. Баумана
- Работал Principal Software Engineer в Oracle
- Старший руководитель разработки реалтайм систем кредитного скоринга
- Консультант
- Люблю Stream Processing и Machine Learning
- Работаю над LLM-системами
- Контрибьютор в Apache Flink

Продакшн в теории и в реальности

kaggle

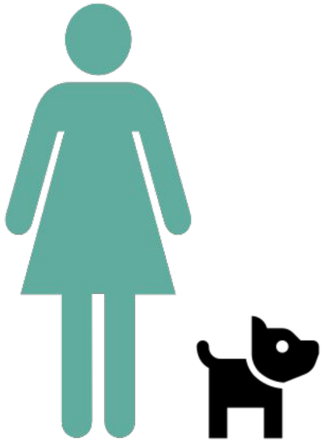


Production



Задача

История



Мария

София

Одобряем?

Наша компания

Custom Dog Portraits on Rocks - Painted Dogs ...

★★★★★ (185)

865.60 SEK

Задача

На основе данных о клиенте мы хотим разработать **статистическую модель** для предсказания **вероятности** того, что клиент не выполнит обязательства по кредиту.

Данные

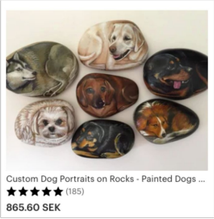
Наши данные

- Покупки
- Баланс
- Споры
- График оплаты
- Невозврат средств

Внешние данные

- Кредитный рейтинг

Данные продавца



- Сумма покупки
- Тип продукта
- Платежный инструмент
- Название продавца

Дизайн системы. Теория

Потоки данных

СЛОЙ РАСЧЕТА ПРИЗНАКОВ В РЕАЛЬНОМ ВРЕМЕНИ

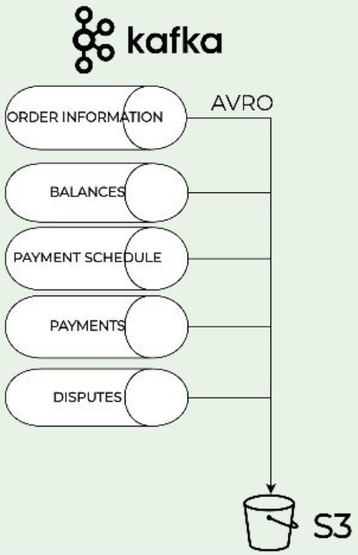
Расчет признаков клиента в момент покупок

СЛОЙ АНАЛИТИКИ

Анализ данных
Конструирование признаков
Тренировка моделей

Потоки данных

РЕАЛТАЙМ

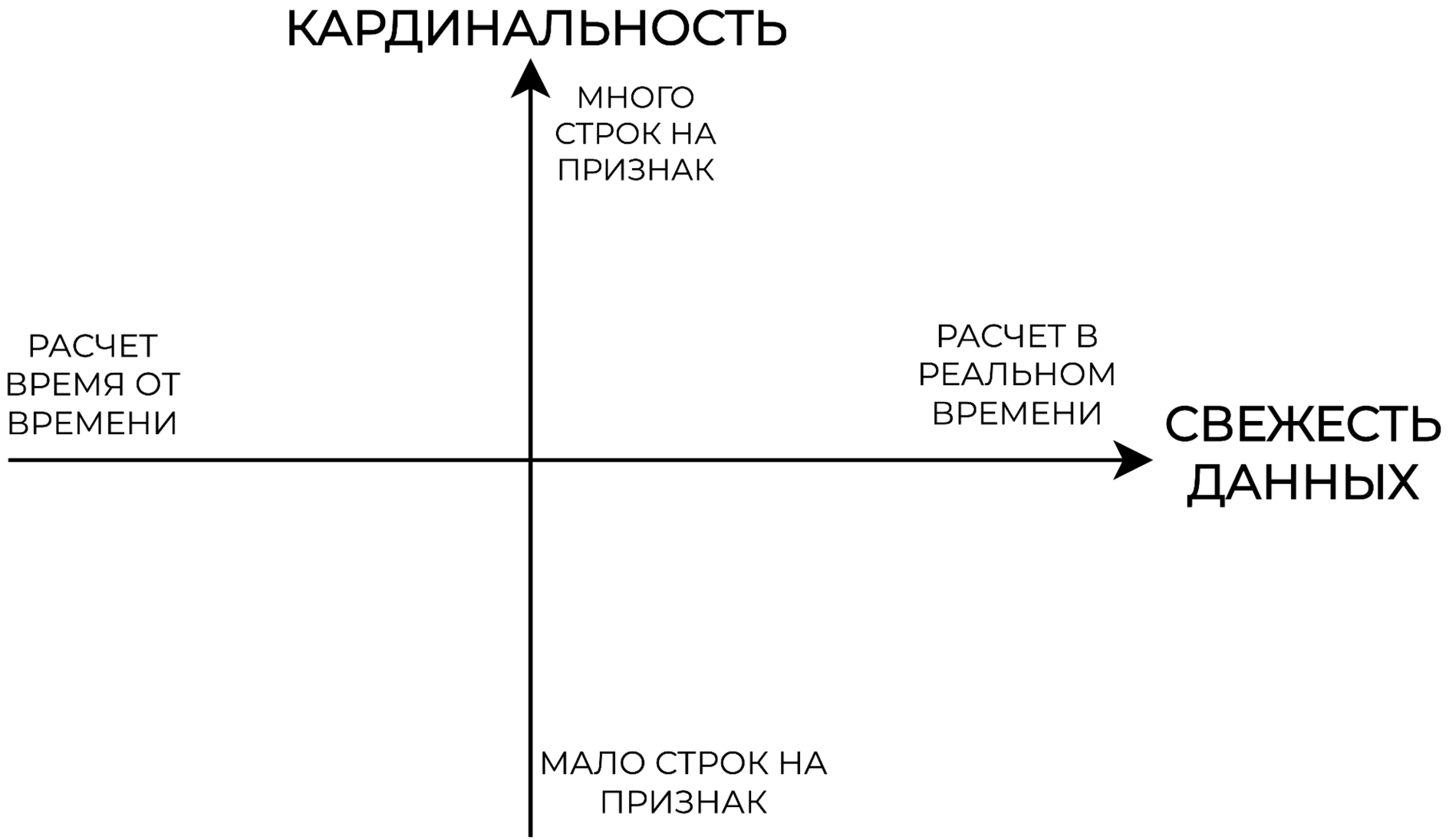


Расчет признаков клиента в момент покупок

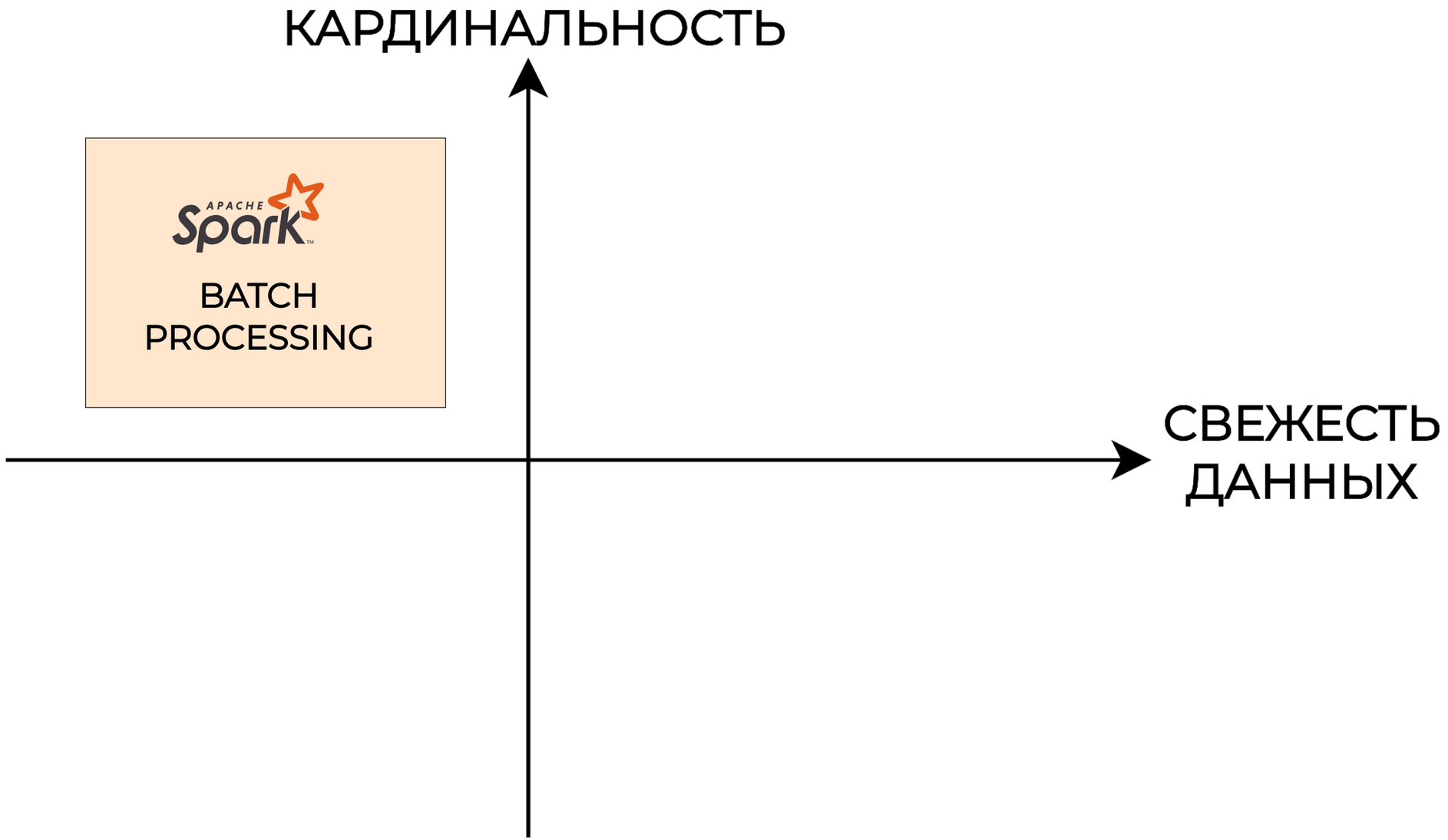
АНАЛИТИКА

Анализ данных
Конструирование признаков
Тренировка моделей

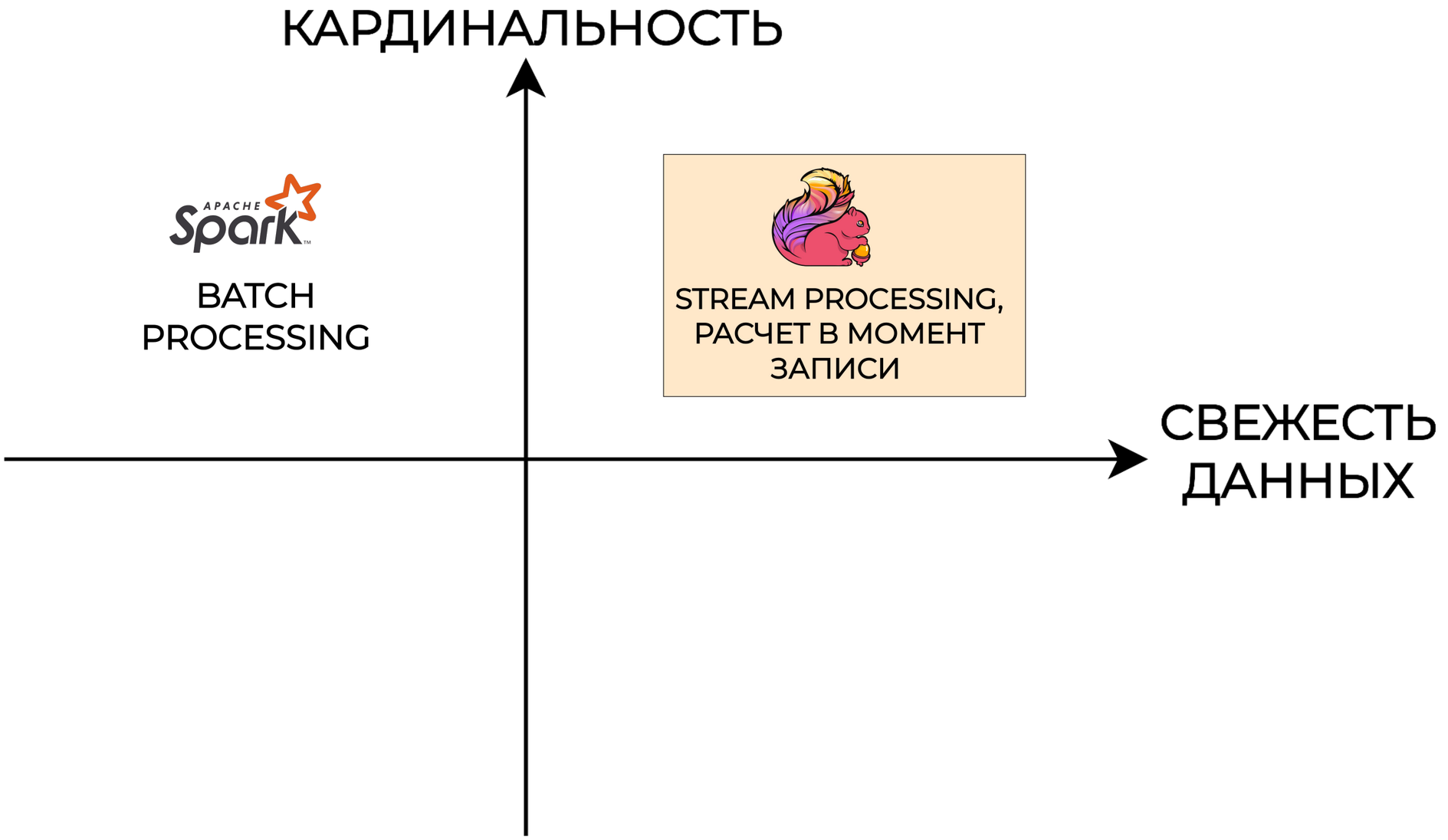
Признаки


КАРДИНАЛЬНОСТЬ
МНОГО СТРОК НА ПРИЗНАК
РАСЧЕТ ВРЕМЯ ОТ ВРЕМЕНИ
РАСЧЕТ В РЕАЛЬНОМ ВРЕМЕНИ
СВЕЖЕСТЬ ДАННЫХ
МАЛО СТРОК НА ПРИЗНАК

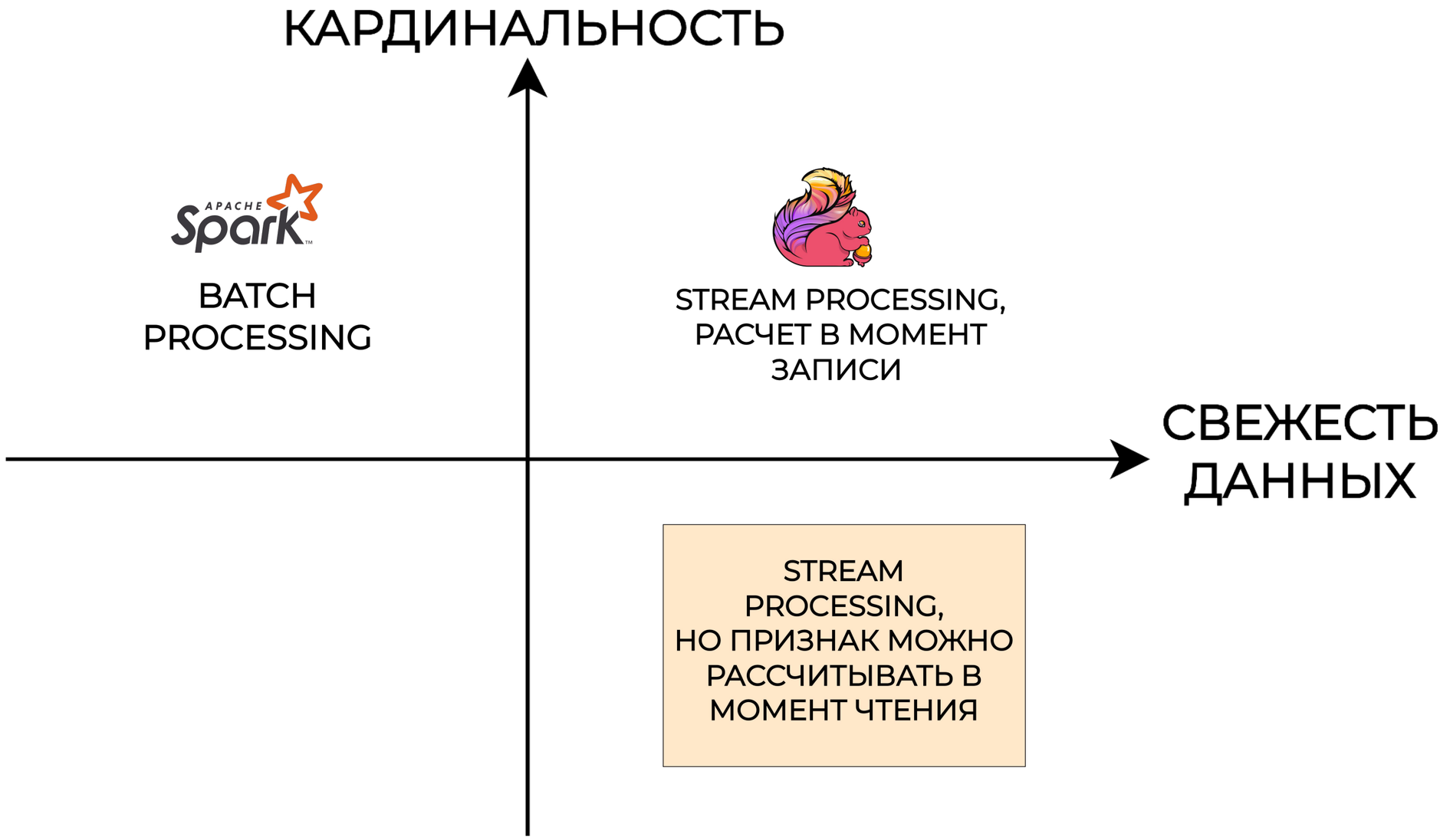
Признаки


КАРДИНАЛЬНОСТЬ
APACHE Spark
BATCH PROCESSING
СВЕЖЕСТЬ ДАННЫХ

Признаки


КАРДИНАЛЬНОСТЬ
APACHE Spark
BATCH PROCESSING
STREAM PROCESSING, РАСЧЕТ В МОМЕНТ ЗАПИСИ
СВЕЖЕСТЬ ДАННЫХ

Признаки


КАРДИНАЛЬНОСТЬ
BATCH PROCESSING
STREAM PROCESSING, РАСЧЕТ В МОМЕНТ ЗАПИСИ
СВЕЖЕСТЬ ДАННЫХ
STREAM PROCESSING, НО ПРИЗНАК МОЖНО РАССЧИТЫВАТЬ В МОМЕНТ ЧТЕНИЯ

Признаки

КАРДИНАЛЬНОСТЬ

BATCH PROCESSING

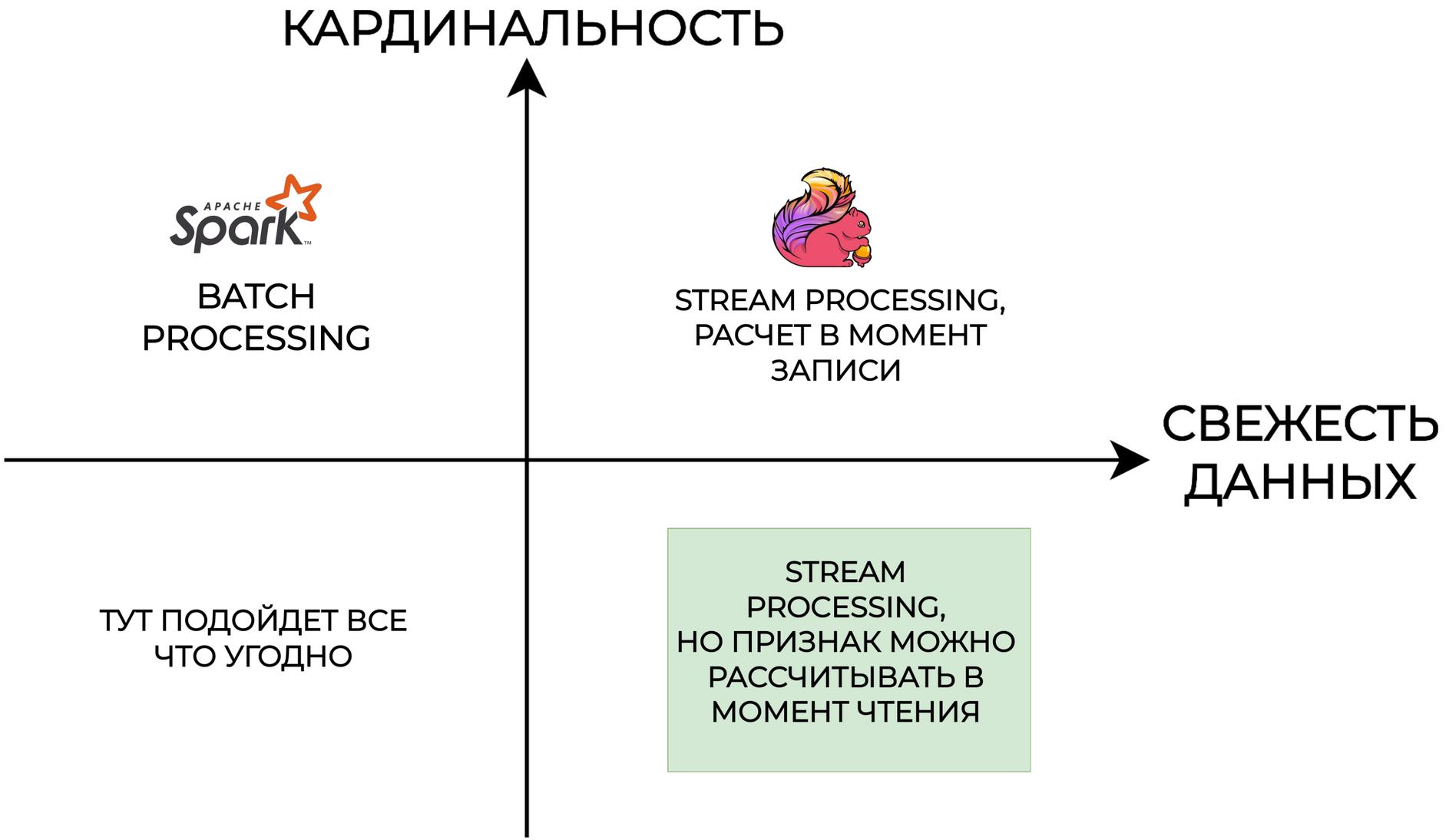
STREAM PROCESSING, РАСЧЕТ В МОМЕНТ ЗАПИСИ

СВЕЖЕСТЬ ДАННЫХ

ТУТ ПОДОЙДЕТ ВСЕ ЧТО УГОДНО

STREAM PROCESSING, НО ПРИЗНАК МОЖНО РАССЧИТЫВАТЬ В МОМЕНТ ЧТЕНИЯ

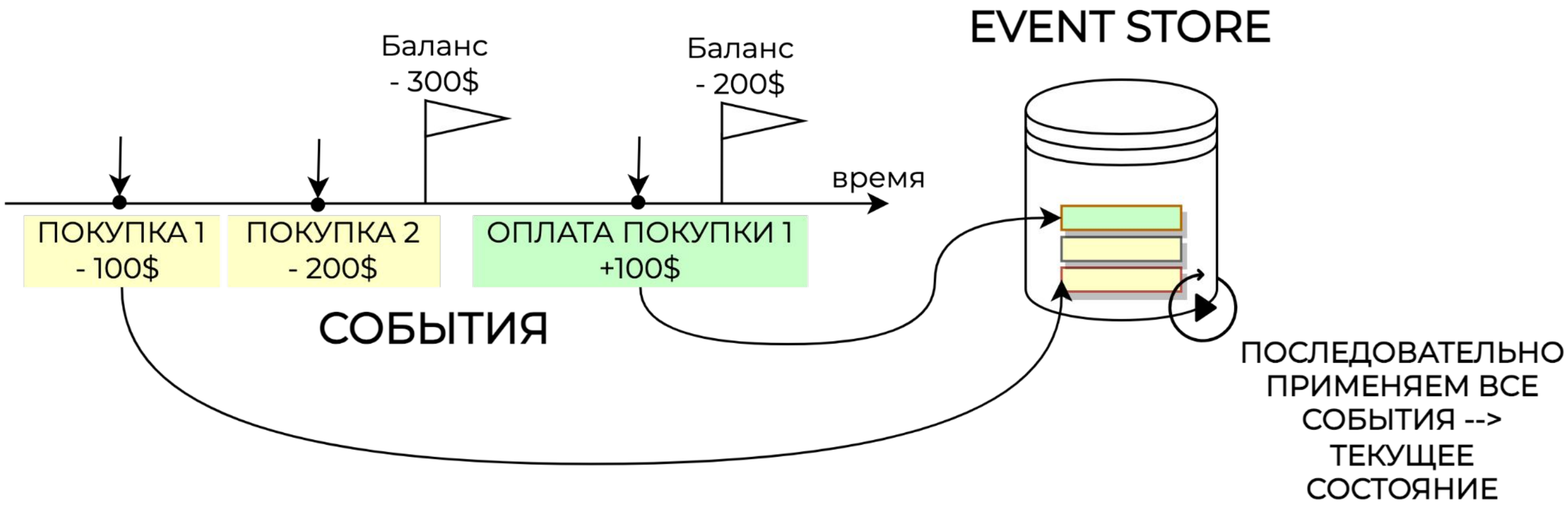
Признаки


КАРДИНАЛЬНОСТЬ
BATCH PROCESSING
STREAM PROCESSING, РАСЧЕТ В МОМЕНТ ЗАПИСИ
СВЕЖЕСТЬ ДАННЫХ
ТУТ ПОДОЙДЕТ ВСЕ ЧТО УГОДНО
STREAM PROCESSING, НО ПРИЗНАК МОЖНО РАССЧИТЫВАТЬ В МОМЕНТ ЧТЕНИЯ

Event sourcing

- Модель, предсказывающая вероятность невыполнения обязательств по кредиту
- Модель надо тренировать на исторических данных
- Надо знать, как менялось наше знание о клиенте в течение времени

Event sourcing



CQRS

ЛЕГКО ЗАПИСЫВАТЬ

EVENT STORE

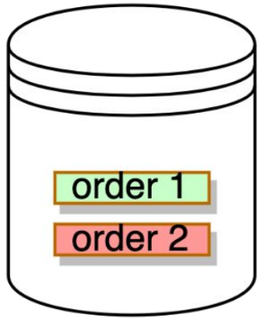
ИСТОРИЯ

Hash Key: Order ID

СИНХРОНИЗАЦИЯ

ЛЕГКО ЧИТАТЬ

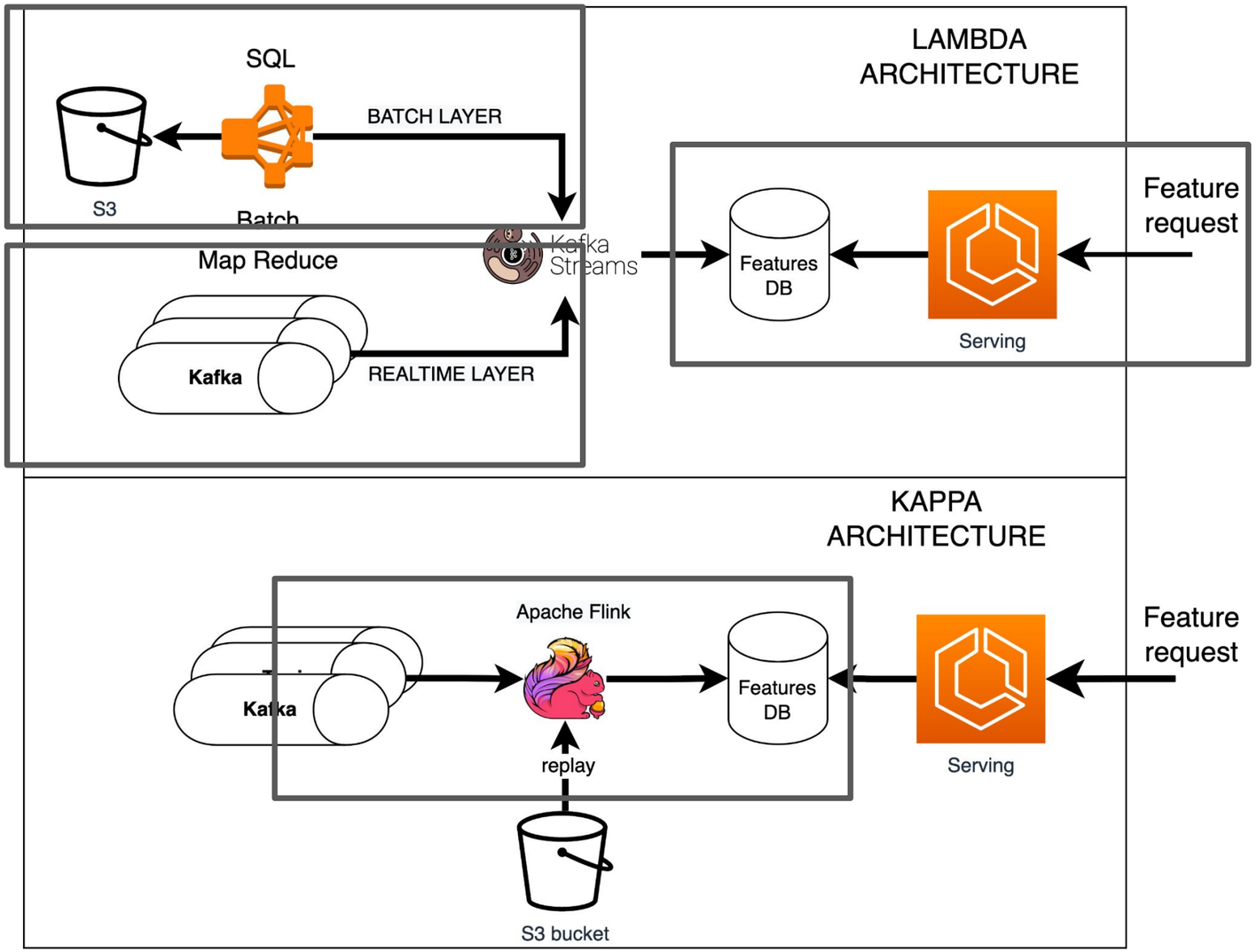
STATE STORE



ТЕКУЩЕЕ СОСТОЯНИЕ КАЖДОЙ ПОКУПКИ

Hash Key: Customer ID

Lambda и Kappa архитектуры

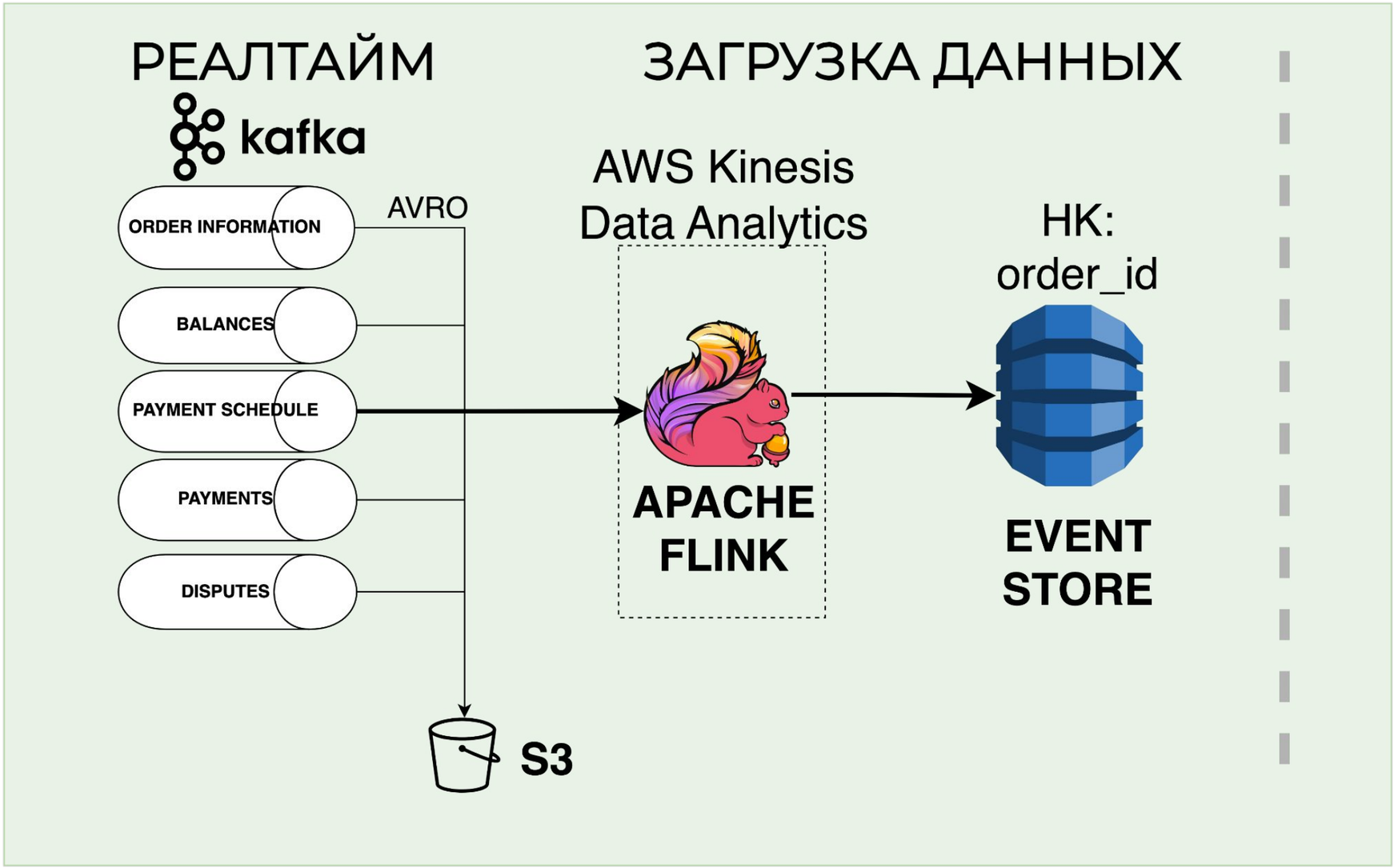


Принятые решения

- Используем Event Sourcing и CQRS
- Выбираем архитектуру Kappa

Дизайн системы. Практика

Модель на запись



Модель на запись

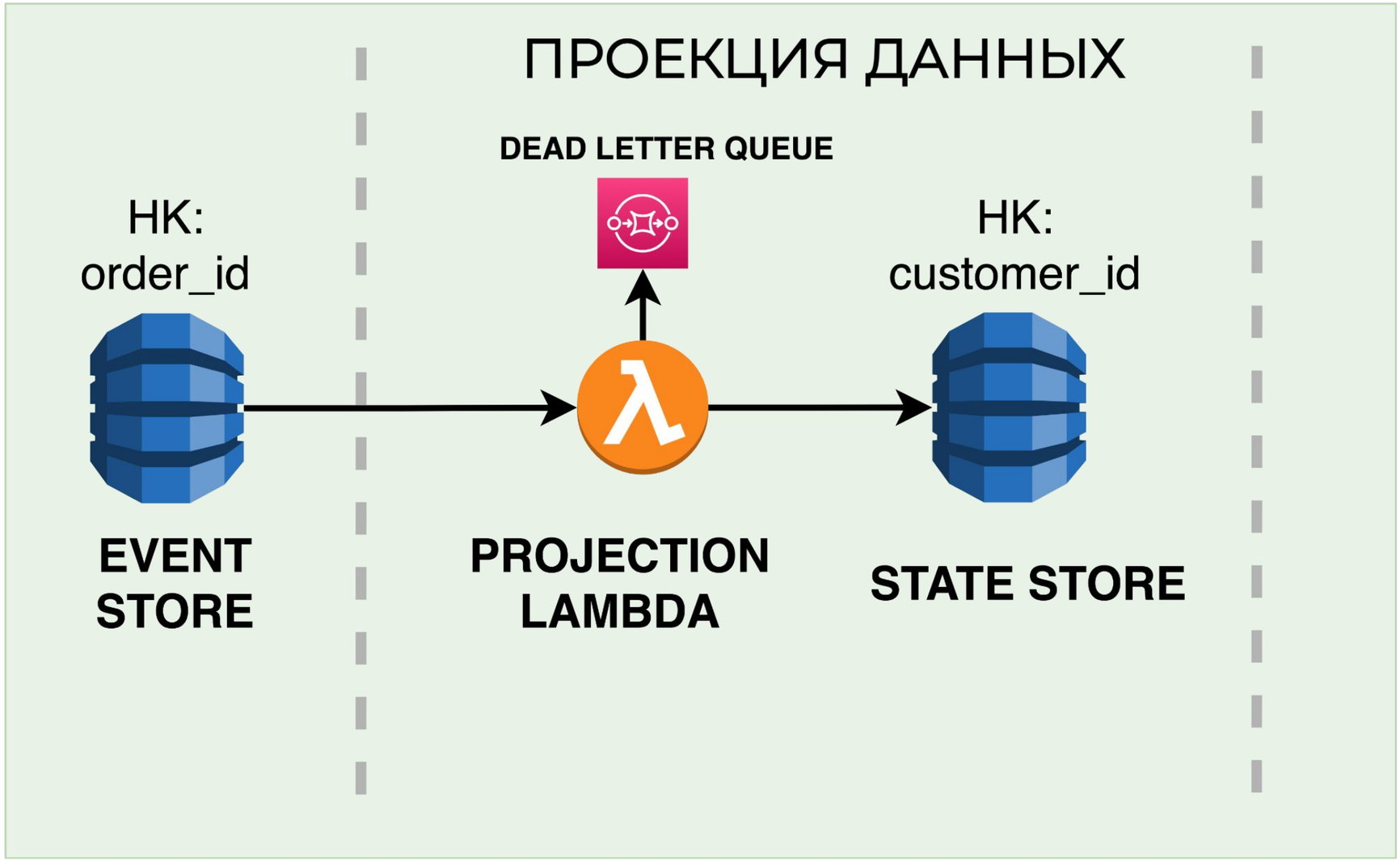
Модель на чтение

```python
@deserialize
@dataclass
class OrderState(AvroModel):
        customer_id: None
        order_information: Optional[OrderInformation] = field(default=None)
        balances: Dict[str, Balance] = field(default_factory=dict)
        payments_schedule: Dict[str, PaymentsSchedule]
                = field(default_factory=dict)
        disputes: Dict[str, Dispute] = field(default_factory=dict)

@deserialize
@dataclass
class Dispute(AvroModel):
        id: str = field()
        balance_id: str = field()
        opened_at: Optional[int] = field(default=None)
        closed_at: Optional[int] = field(default=None)
        reason: Optional[str] = field(default=None)
```

Модель данных. Обработчики сообщений

```
def handle_dispute_opened_event(arguments: Arguments) -> OrderState:
        dispute_id: str = arguments.event['dispute_id']
        balance_id: str = arguments.event['balance_id']
        opened_at: int = int(arguments.event['opened_at'])

        dispute: Dispute = Dispute(
                dispute_id=dispute_id,
                balance_id=balance_id,
                reason=arguments.event['reason'],
                opened_at=opened_at
        )

        arguments.state.disputes[dispute_id] = dispute

        return arguments.state

def handle_dispute_closed_event(arguments: Arguments) -> OrderState:
        dispute_id: str = arguments.event['dispute_id']
        dispute: Dispute = arguments.state.disputes.get(dispute_id)
        dispute.closed_at = int(arguments.event['closed_at'])
        arguments.state.disputes[dispute_id] = dispute

        return arguments.state
```

Синхронизация

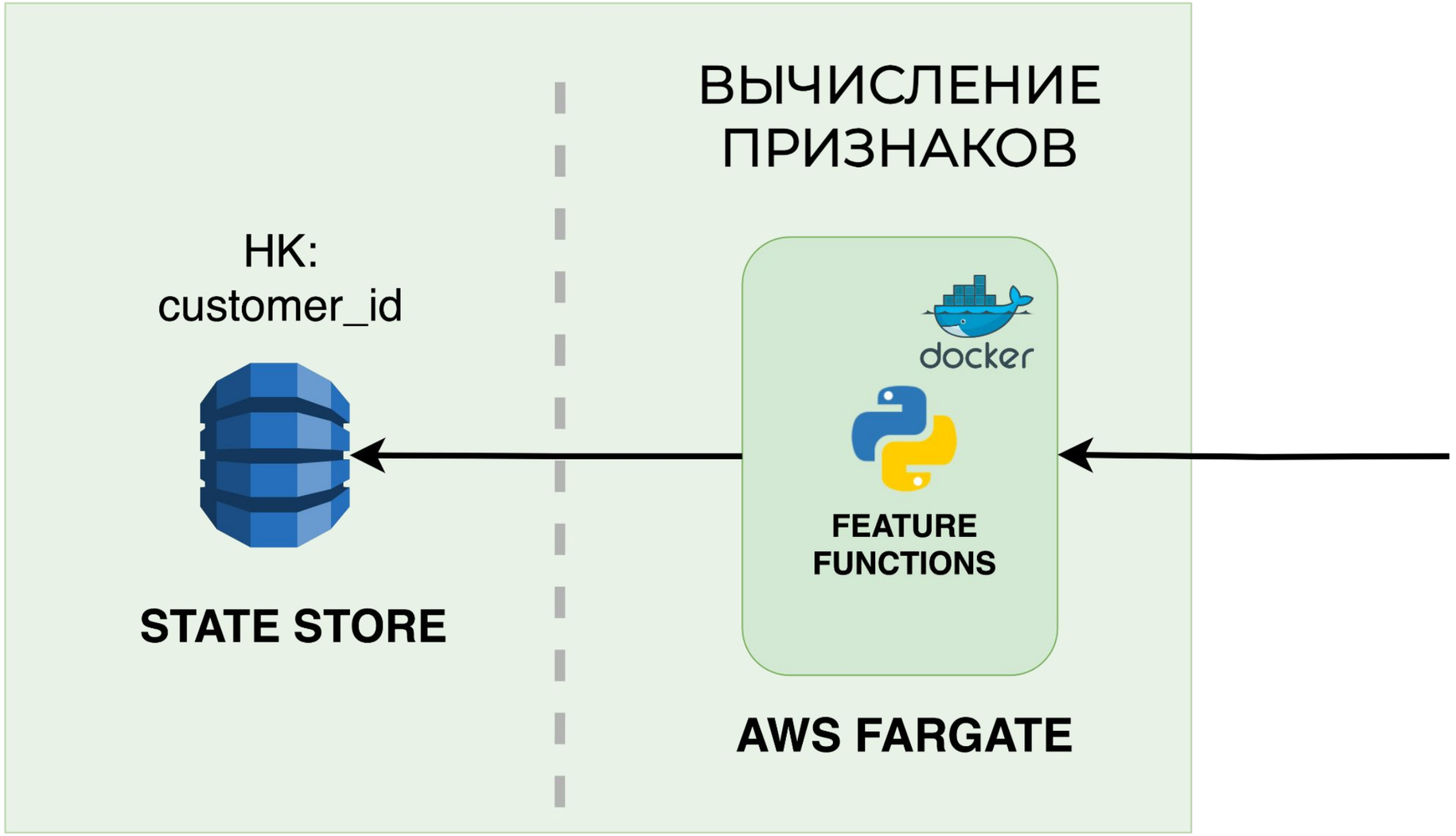


Модель на чтение

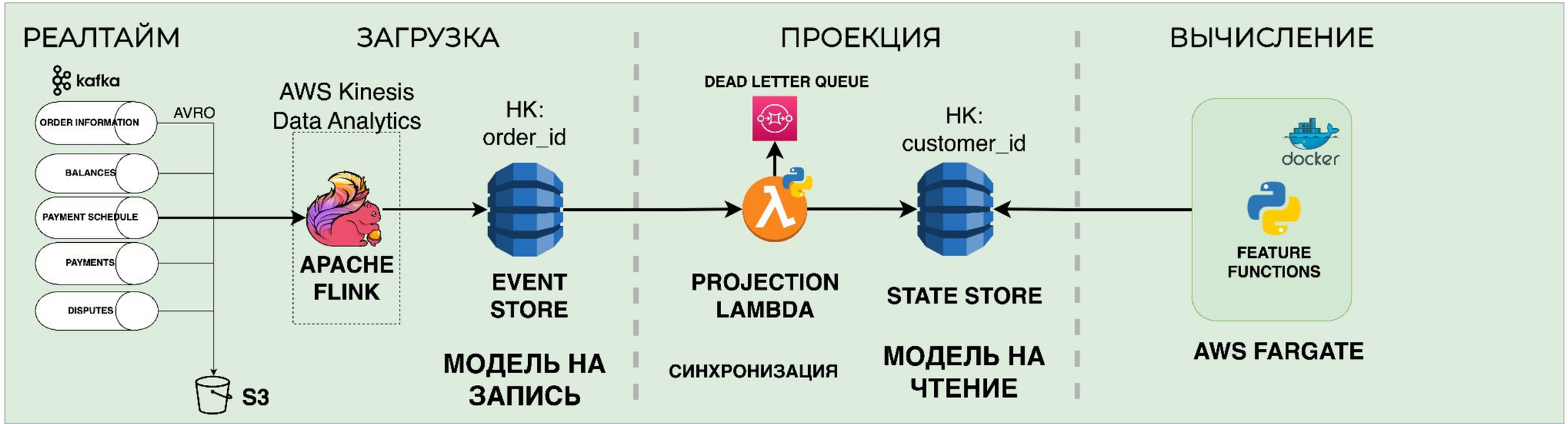
Вычисление признаков

```python
class CountOrdersPaid(IStatefulFeature):

@classmethod
def get_name(cls) -> str:
        return "count_orders_paid"

@classmethod
def calculate(cls, projections: List[OrderState], decision_time: Optional[int] = None) -> int:
        num_orders_paid: int = 0
        for projection in projections:
                if is_order_paid(projection):
                        num_orders_paid += 1

        return num_orders_paid

@classmethod
def get_description(cls) -> str:
        return """
                Count the number of orders that are fully paid
        """
```

Вычисление признаков



Общая архитектура



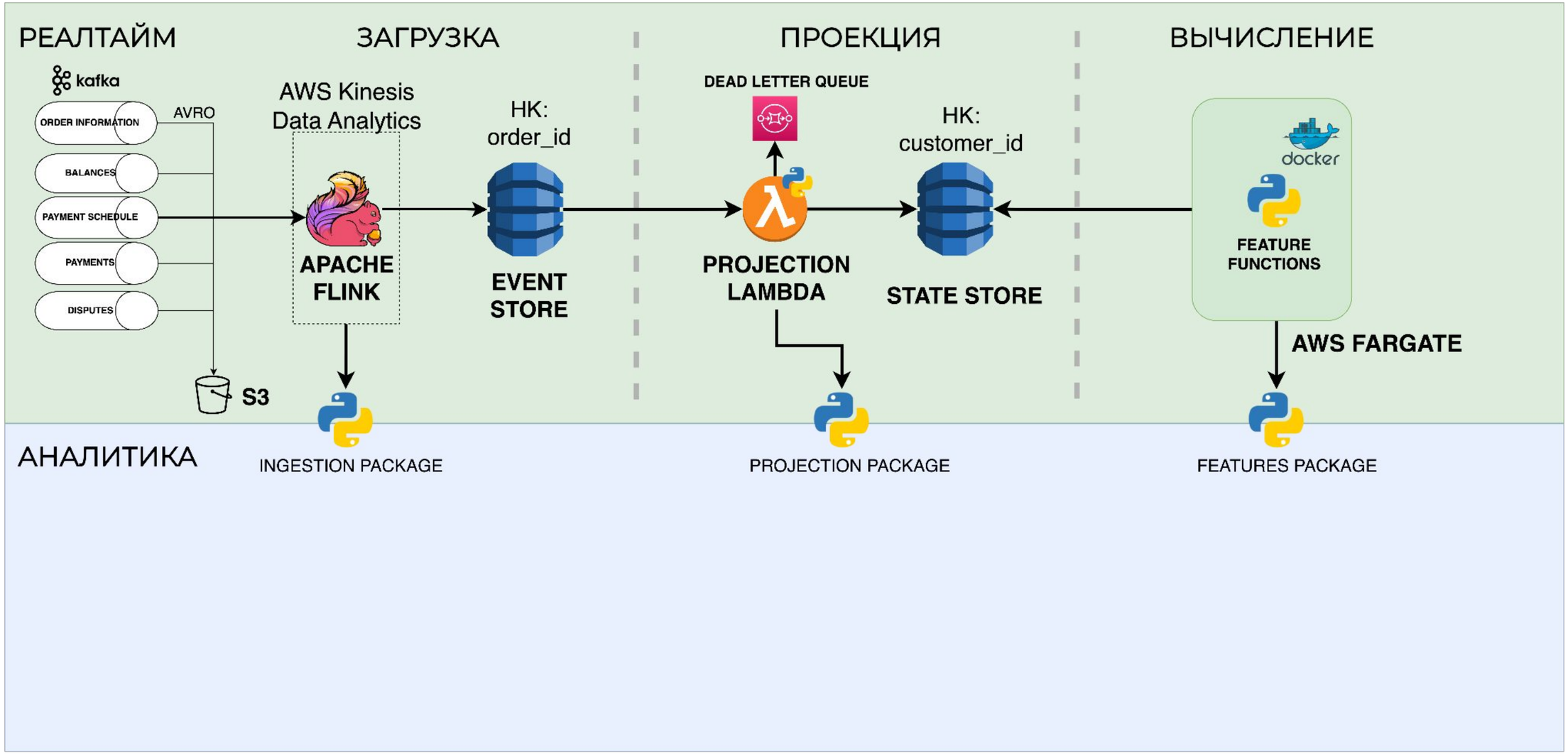
АНАЛИТИКА

Анализ данных
Конструирование признаков
Тренировка моделей

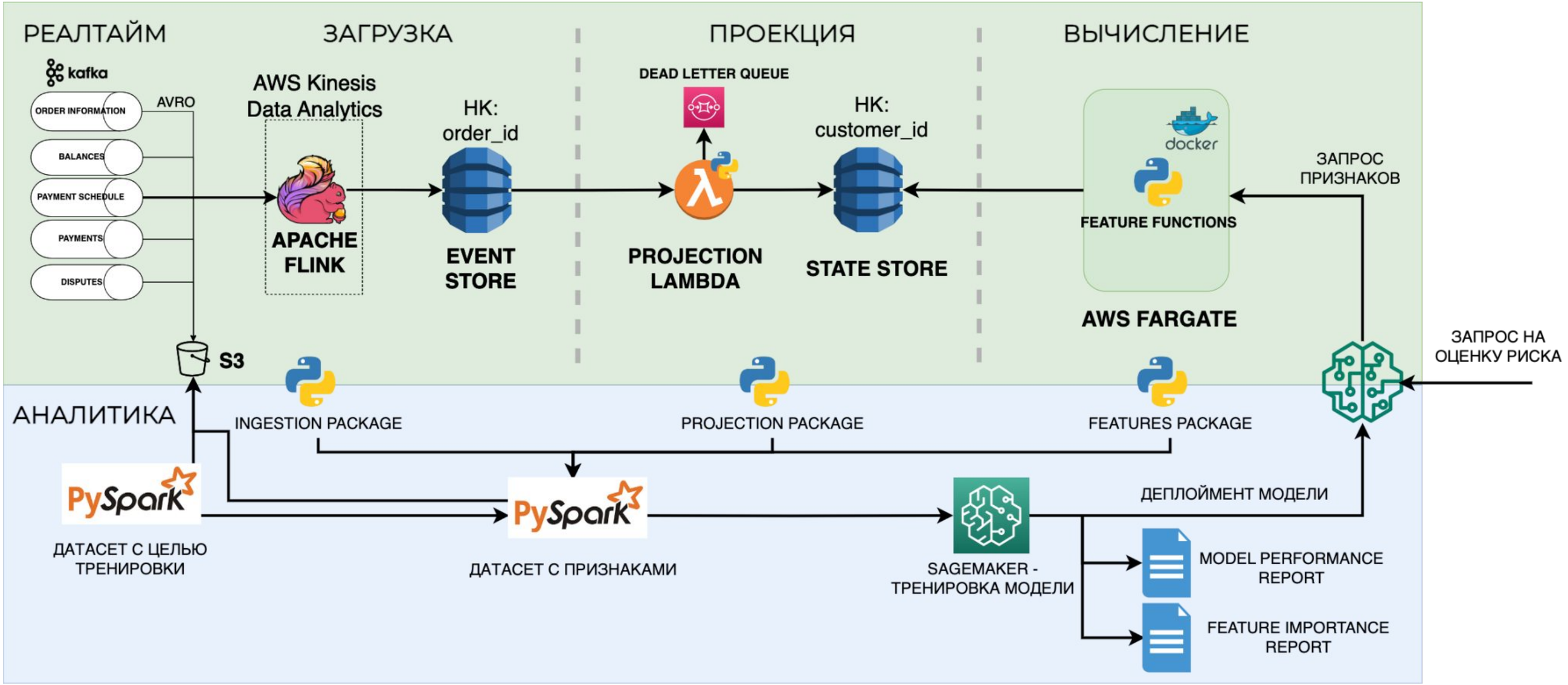
Слой аналитики

- Генерируем датасет для обучения модели. Были ли оплачены покупки?
- Вычисляем признаки для временных меток, в которые были сделаны покупки
- Запускаем обучение модели
- Оцениваем значимость признаков для модели

Слой аналитики



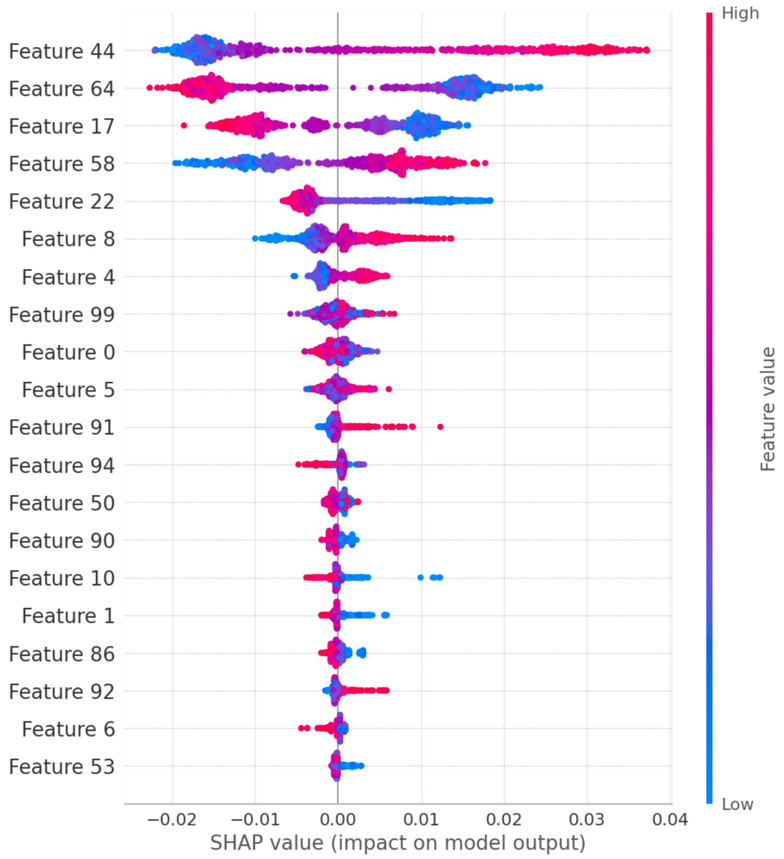
Слой аналитики



Так в чем секрет?

- Создаем и проверяем значимость признаков в офлайн
- Когда нас все устраивает, добавляем в реалтайм
- Новые признаки создавать просто
- Чаще всего не нужно пересчитывать данные всех клиентов
- Но при необходимости для этого можно использовать слой аналитики

Важность признака для модели (SHAP Value)




The End