Фреймворк для AI/ML-проектов

(R&D-версия)

Описание архитектуры

Skoltech

Веб-страница проекта: http://dataskai.com

Содержание

Перечень принятых сокращений

Big Data	— совокупность подходов, инструментов и методов обработки структурированных и неструктурированных данных значительных объемов и многообразия для получения воспринимаемых человеком результатов, эффективных в условиях непрерывного прироста
Data Mining	— совокупность методов обнаружения в данных ранее неизвестных, нетривиальных, практически полезных и доступных интерпретации знаний, необходимых для принятия решений
IoT	— интернет вещей, методология вычислительной сети физических предметов, оснащенных встроенными технологиями для взаимодействия друг с другом или с внешней средой, рассматривающая организацию таких сетей как явление, способное перестроить экономические и общественные процессы, исключающее из части действий и операций необходимость участия человека.
Predictive Quality	— активное выявление признаков ошибок с целью повышения продуктивности ресурсов и качества продукта.
SAP ERP	— система управления ресурсами предприятия компании SAP SE.
Машинное обучение	— математическая дисциплина (раздел искусственного интеллекта), использующая разделы математической статистики, численных методов оптимизации, теории вероятности, дискретного анализа, и извлекающая знания из данных.
Обучение без учителя	— один из способов машинного обучения, который в процессе наблюдения за системой строит модель нормального (штатного) функционирования системы и позволяет определять отклонения от нормы в новых наблюдениях.
ОПО	— общесистемное программное обеспечение.
ПКИД	— программный комплекс интеллектуальной диагностики.
ПО	— программное обеспечение.
Предиктивный анализ	— класс методов анализа данных, концентрирующийся на прогнозировании будущего поведения объектов и субъектов с целью принятия оптимальных решений.
Предсказательные модели	— математические модели, построенные по данным эксплуатации технических систем, позволяющие предсказать их поведение. Например, математическая модель может быть построена для предсказания с требуемой точностью отклонения поведения системы от нормального, детектирования разладки в поведении системы, классификации типов разладки и т.д.
ПЭВМ	— персональная электронно-вычислительная машина (персональный компьютер).
СПО	— специальное программное обеспечение.
ОПО	— общее программное обеспечение.

1 Круг задач, решаемых с помощью фреймворка

Фреймворк DATASKAI (далее — DATASKAI, фреймворк) для автоматизации сбора, хранения, обработки и анализа данных для Data Science решений. С его помощью возможна автоматизация и упрощение постановки задач, создание и оценка моделей машинного обучения и внедрение моделей в рабочие процессы.

В частности, **DATASKAI позволяет:**

– собирать данные из внешних информационных систем и преобразовать их для эффективного хранения, обработки и анализа;

– реализовывать унифицированный доступ к данным к данным через внешние интерфейсы; обрабатывать данные, встраивать в систему экспертные модели предметной области, делать предсказания на данных;

– формализовывать Data Science задачи, проводить анализ данных и оценивать полученные решения.

В первую очередь, DATASKAI ориентирована на работу с данными, зависящими от времени, но её функциональность может быть расширена с помощью открытого API.

Примеры задач, которые можно решить с помощью DATASKAI:

– оптимизация технологических процессов непрерывного производства продукции;

– предсказательное техническое обслуживание сложных изделий;

– прогнозирование оттока клиентов организации.

2 Общие характеристики

2.1 Языки программирования и парадигма построения комплекса

Основные языки разработки DATASKAI – Java и Python. Инфраструктура в основном построена на Java-компонентах для повышения производительности и масштабируемости, а компонент, отвечающий за аналитику и решение бизнес-задач, находится на Python.

Работа над одной предметной областью подразумевает развертывание отдельного экземпляра программного пакета, иными словами, если есть несколько предметных областей, необходимо развернуть несколько экземпляров комплекса, которые обеспечивают работу с ними. Но в этом случае дублирование некоторых подсистем не является необходимым, например, вы можете использовать одно общее хранилище данных.

Компоненты Java встроены в парадигму микросервисной архитектуры с использованием Java 8, среды Spring Boot и дополнительных библиотек для работы с базами данных, очередями сообщений, внешними протоколами и т. Д. Для построения проектов используется Gradle 4.10.3.

Каждый компонент службы запускается в виде контейнера Docker и объединяется с другими службами в виде отдельного контура. Создание контуров выполняется с помощью Docker Compose. Компоненты Python также упакованы и объединены в соответствии с их конфигурациями. Сборка, тестирование, контейнеризация и развертывание осуществляются с помощью GitLab CI / CD.

2.2 Версии языков программирования

При разработке ПК использованы языки высокого уровня:

— Python 3.6;

— Java 8.

2.3 Программное обеспечение, необходимое для функционирования программного продукта

Продукт поддерживает функционирование под управлением операционных систем семейства Linux, выбранных исходя из наибольшей распространенности среди серверных ОС. При развертывании DATASKAI на наборе серверов для связи между модулями используется стек протоколов TCP/IP.

При реализации DATASKAI используются языки программирования высокого уровня Python и Java.

Развертывание компонентов ПК осуществляется преимущественно с использованием технологий контейнеризации, таких как Docker, Kubernetes, Singularity. Преимуществами такого подхода — высокий уровень автоматизации процедуры развертывания, удобство переносимости решения, эффективное использование аппаратной инфраструктуры.

В качестве базовой технологии для реализации подсистемы сбора и импорта данных выбрана технология Kafka.

Для реализации *подсистемы хранения данных* выбраны следующие технологии, ориентированные на хранение различных типов данных:

— для хранения структурированных данных с возможностью эффективного доступа к отдельным записям и их атрибутам с помощью языка SQL используется реляционная СУБД PostgreSQL. При наличии больших объемов данных возможно использование NoSQL-хранилищ, таких как Elasticsearch, MongoDB и Cassandra;

— для хранения временных рядов с высоким разрешением и поддержкой эффективного доступа к таким данным используются такие решения, как TimescaleDB и OpenTSDB;

— для хранения данных в виде файлов, включая неструктурированные и большие массивы данных, используется распределенная файловая система HDFS.

Выбор данных решений обусловлен их уровнем развития и распространением на условиях открытого исходного кода.

Для управления вычислительной инфраструктурой и запуска заданий используются существующие решения в виде Hadoop YARN и Kubernetes.

Для организации масштабируемой распределенной обработки данных большого объема в рамках подсистемы обработки данных ПК используются Hadoop и Spark, как наиболее развитые открытые решения в данной области.

Для реализации подсистемы мониторинга данных и объектов предметной области используются открытые решения Graphite (сбор и хранение метрик) и Grafana (визуализация и оповещения).

3 Область применения

Продукт может применяться для решения описанных задач компаниями (организациями) в целях оптимизации бизнес- и производственных процессов, характеризуемых генерацией потоков данных.

Примеры конкретных областей применения программного продукта:

— задачи оптимизации технологических процессов непрерывного производства в рамках организации, осуществляющей изготовление продукции;

— задачи предсказательного техобслуживания в рамках организации, осуществляющей техобслуживание и ремонт сложных технических изделий;

— задачи прогнозирования оттока клиентов организации, осуществляющей оказание телекоммуникационных услуг.

4 Классы решаемых задач

Основные задачи обучения с учителем, решаемые платформой:

1) классификация — задача определения принадлежности объекта к одному из заранее заданных классов объектов;

2) регрессия — задача определения скалярной или векторной характеристики объекта;

3) прогнозирование — задачи классификации и регрессии с использованием исходных данных в виде временных рядов;

4) ранжирование — задача определения правильного порядка расстановки объектов (может быть также сведена к 1 или 2);

5) рекомендация — задача имеет множество постановок, однако решаема при декомпозиции и сведении к набору из задач 1, 2, 3 + задача оптимизации.

Основные задачи обучения без учителя, решаемые платформой:

1) кластеризация — задача определения подвыборок (кластеров) объектов и алгоритма, с помощью которого можно разбить исходную выборку на подвыборки;

2) anomaly/novelty detection — задача определения аномальных объектов в выборке объектов;

3) change detection — задача определения точки изменения в упорядоченном ряде объектов.

Также DATASKAI позволяет проводить коллаборативную работу над задачами на данных, не укладываемых в парадигму ML. К данному пункту относятся задачи, решения которых можно проверить, имея данные ответов и составив специальные метрики. Однако алгоритм их решения тяжело или невозможно получить через решение ML-задач. Примерами таких задач может служить разработка и улучшение существующих (в определенных тестовых сценариях) алгоритмов, решающих NP-полные задачи (задача о рюкзаке, коммивояжера и т.п.) путем применения дополнительных эвристик.

5 Архитектура программного продукта

Реализация архитектуры продукта предусматривает:

— наличие инфраструктуры из существующих информационных систем (далее – ИС), являющихся источниками данных;

— реализацию технических решений для обмена данными между платформой и ИС и внедряемыми новыми объектами архитектуры, совместно обеспечивающих возможность обработки данных.

Архитектура DATASKAI включает в себя следующие уровни, изображенные на рис. 1:

— подсистема сбора данных (Data Ingestion Subsystem). На этом уровне данные собираются, доставляются и преобразуются из внешних источников и загружаются на следующий уровень;

— подсистема хранения данных (Data Storage Subsystem). Используется для хранения и учета загруженных данных и результатов их последующей обработки. Подсистема также обеспечивает доступ к данным для компонентов DATASKAI;

– подсистема обработки данных (Data Processing Subsystem). Это масштабируемая вычислительная инфраструктура для выполнения обработки данных по запросу компонентов системы. Также на этом уровне пользователи должны описать предметную область - абстракцию предметной области для вышележащих слоев;

– подсистема анализа данных (Data Analysis Subsystem). Включает в себя инструменты для анализа данных, состояний объектов домена и их атрибутов;

– пользовательские инструменты и расширения (User tools and extensions). Созданы для работы с администраторами DATASKAI и целевыми пользователями (исследователями данных, аналитиками, экспертами отрасли, менеджерами). Этот уровень включает инструменты для Data Science (см. Data Science Toolkit), визуализирующие данные через графический интерфейс, выводящие аналитику и прогнозы для помощи в принятии решений (например, управление производственным процессом, составление выводов о необходимости ремонта или замены устройства), мониторинг состояния промышленного объекта и т. д.);

– сервисные подсистемы (Service subsystems). Реализуют функции мониторинга, безопасности и др.

Компонентами DATASKAI являются:

– инструменты оценки (Evaluation tools). Предоставляют набор инструментов data science;

– инструментарий CLI (CLI Toolkit). Инструменты для облегчения обработки данных DATASKAI;

– веб-приложение для сабмитов (рассмотрения предложений) (Submits Web App). Отображает графический интерфейс рейтинга для предложений;

– сервис метрик (Metrics Service). Вычисляем значения метрик для сабмитов.

– мастер развертывания проекта DATASKAI (DATASKAI Deploy Project Wizard);

– построитель прогнозов (Prediction Builder). Автоматизирует процесс вывода через REST API;

– компонент Feature Store. Предоставляет REST API для работы с функциями;

– проигрыватель моделей (Models Player). Автоматизирует жизненный цикл модели машинного обучения;

– упаковщик моделей (Model Wrapper). Упаковывает модель машинного обучения для совместимости с Models Player;

– построитель отличительных признаков (Feature Builder). Автоматизирует процесса построения отличительных признаков через REST API;

– служба данных (Data Service). Предоставляет REST API для запросов к хранилищу данных.

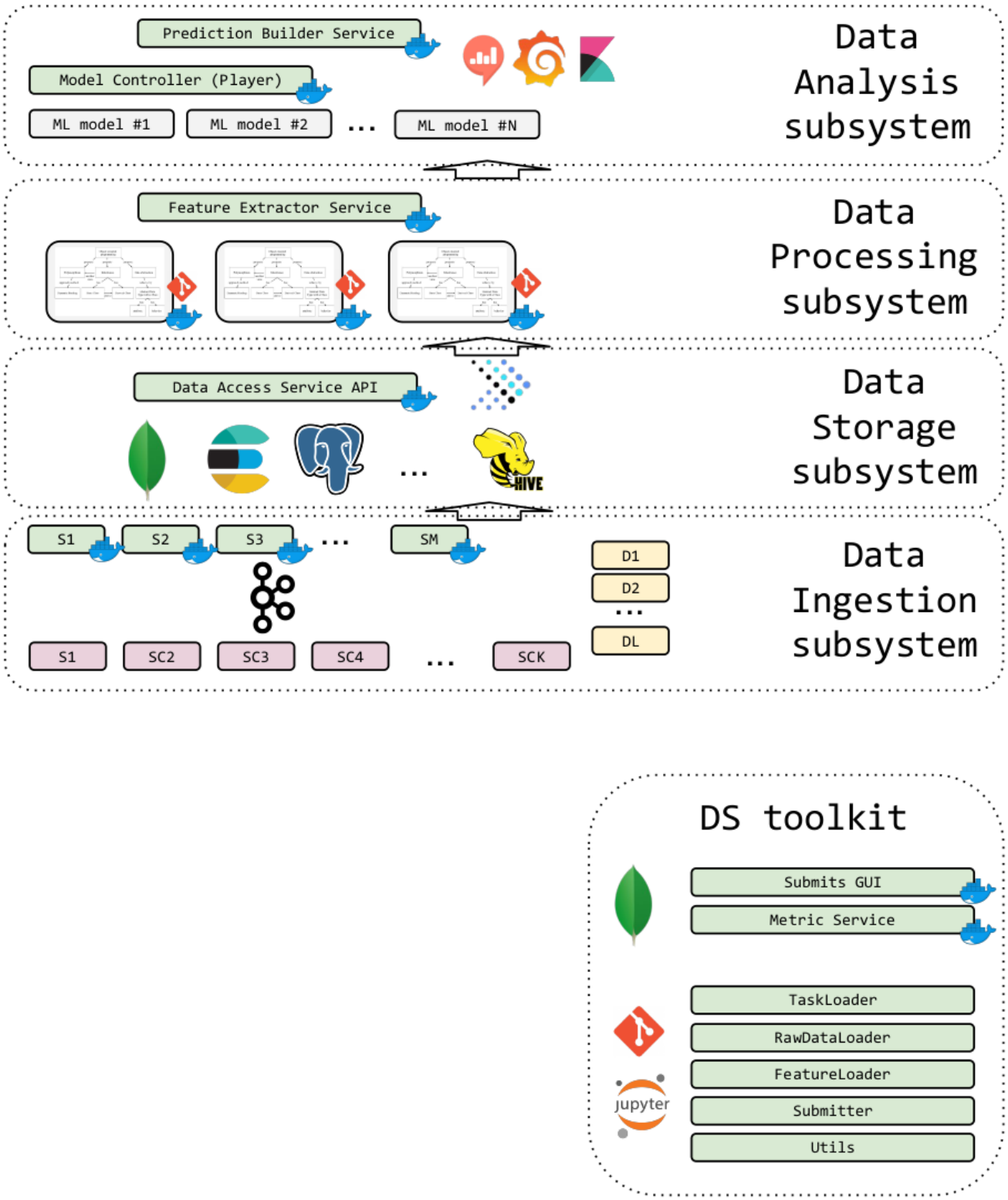

Рис. 1. Архитектура программного комплекса DATASKAI

6 Структура программного продукта с описанием функций составных частей и связи между ними

По языкам, использующимся в продукте, его можно условно разделить на две части:

— Java—компоненты;

— Python—компоненты.

Инфраструктурная составляющая ПК в основном построена на Java-компонентах для обеспечения производительности и масштабируемости, а составляющая, отвечающая за работу аналитики и решение конкретной бизнес-задачи, построена на языке Python.

Работа над одной предметной областью подразумевает развертывание экземпляра DATASKAI целиком, иными словами, при наличии нескольких предметных областей необходимо производить развертывание нескольких экземпляров DATASKAI, обеспечивающих работу с ними. Стоит отметить, что это не всегда ведет к развертыванию дополнительных нижележащих систем, например хранилищ данных.

Java—компоненты построены в парадигме микросервисной архитектуры с использованием Java 8, фреймворка Spring Boot, а также с дополнительными библиотеками для работы БД, очередями сообщений, внешними протоколами и т.д. Для сборки проектов используется Gradle 4.10.3.

Каждый сервис компонента запускается в виде Docker—контейнера и объединяется с остальными сервисами в виде отдельного контура. Создание контура осуществляется с помощью Docker Compose. Сборка, тестирование, контейнеризация и развертывание сервисов обеспечиваются средствами GitLab CI/CD.

Python—компоненты запускаются в виде Docker—контейнеров и объединяются с остальными согласно своим конфигурациям. Сборка, тестирование, контейнеризация и развертывание Python—компонентов обеспечивается средствами GitLab CI/CD.

Возможна поставка продукта в виде, допускающем развертывание в изолированной среде.

7 Подсистемы DATASKAI

7.1 Data Ingestion Subsystem

Подсистема Data Ingestion осуществляет сбор и импорт данных из внешних источников в DATASKAI. Процесс включает опрос источников данных или получение данных от внешних поставщиков, анализ и проверку данных, преобразование данных во внутренние форматы и загрузку данных в подсистему хранения данных.

Компоненты подсистемы:

– Шина данных Data Bus

– Шлюз данных Data Gateway

– Входные соединители Source connectors

– Процессоры данных Data processors

– Соединители приемников Sink connectors

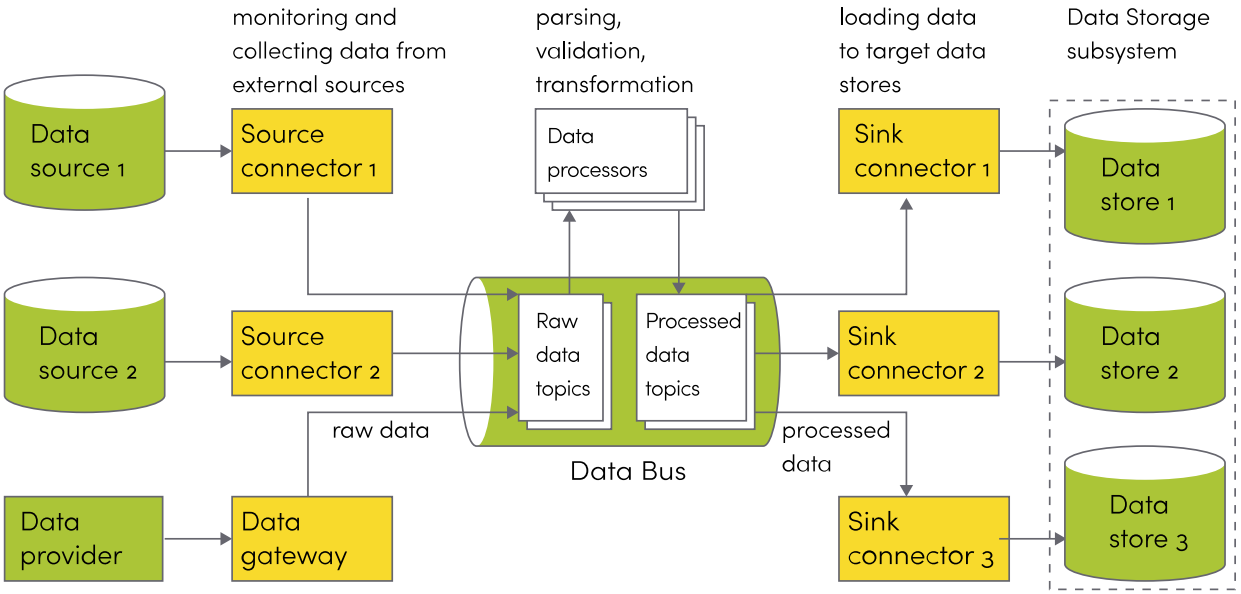

Рис. 2. Схема подсистемы ввода данных Data Ingestion Subsystem

Шина данных Data Bus

Шина данных хранит потоки данных и поддерживает обмен данными между другими частями подсистемы. Шина данных хранит потоки данных записей, организованных по темам, отказоустойчивым и надежным способом в течение настроенного периода хранения. Шина данных позволяет своим клиентам записывать и считывать потоки данных, используя модель производителя/потребителя.

Шина данных - это внешняя зависимость, которая реализуется Apache Kafka, платформой распределенной потоковой передачи с открытым исходным кодом.

Шлюз данных Data Gateway

Data Gateway реализует веб-сервис с REST API для получения данных от внешних поставщиков данных. Шлюз используется в случаях, когда сбор данных должен контролироваться внешней системой, или невозможно реализовать исходный соединитель в Java. Шлюз пересылает полученные данные в шину данных.

Data Gateway - это внешняя зависимость, которая реализуется Confluent REST Proxy, службой RESTful HTTP для взаимодействия с кластерами Kafka.

Входные соединители Source connectors

Входной соединитель осуществляет мониторинг и сбор данных из определенного источника данных. Это подключаемая внешняя зависимость, реализация которой зависит от источника данных. Например, мониторинг источника данных может быть реализован путем опроса или подписки на уведомления. Соединитель направляет собранные данные в шину данных.

Основная задача соединителя источника - надежно передавать необработанные данные в шину данных. Соединитель источника не должен выполнять сложные преобразования данных, чтобы избежать потери исходных данных из-за ошибок анализа. Как только необработанные данные сохраняются в шине данных, они могут быть дополнительно обработаны и преобразованы восстанавливаемым способом с использованием процессоров данных. Как правило, исходный соединитель должен поддерживать как массовую загрузку исторических данных, так и добавочную загрузку последних данных, сохраняя при этом свое состояние между перезапусками. Соединитель гарантирует, что исходные данные не теряются, но возможны дубликаты на стороне шины данных.

DATASKAI не предоставляет готовых соединителей источника, но вместо этого поддерживает использование сторонних реализаций, нацеленных на разные типы источников данных (например, файловые серверы, реляционные базы данных, очереди сообщений). Можно использовать существующие общие коннекторы для Kafka (например, опубликованные на Confluent Hub) или разработать собственные коннекторы. Рекомендуемый способ реализации исходного соединителя - использовать API Java и Kafka Connect. Когда это невозможно, исходный соединитель может быть реализован с использованием любого языка программирования с доступной клиентской библиотекой Kafka.

Процессоры данных Data processors

Процессор данных реализует некоторую логику обработки данных, включая анализ, проверку и преобразование. Основная задача обработчиков данных - преобразовать необработанные данные, предоставленные соединителями источника, в формат, подходящий для загрузки в хранилище данных. Обработчи-

ки данных считывают потоки данных и создают новые потоки данных. Поток входных данных может быть необработанными данными, загруженными соединителем источника или выходными данными другого процессора данных, то есть процессоры данных могут быть соединены в конвейер.

Поскольку логика обработки данных обычно очень специфична для предметной области, DATASKAI не предоставляет готовые процессоры данных, а поддерживает интеграцию с произвольными реализациями процессоров. Реализация процессора данных должна считывать данные из раздела шины данных, применять необходимую логику обработки к каждой записи и записывать выходные записи в другой раздел шины данных. Рекомендуемый способ реализации процессора данных - использовать библиотеки Java / Scala и Kafka Streams. Когда это невозможно, процессор данных может быть реализован с использованием любого языка программирования с доступной клиентской библиотекой Kafka.

Соединители приемников Sink connectors

Соединитель приемника осуществляет загрузку данных из шины данных в хранилище данных, которое является заключительным этапом конвейера приема данных. Соединитель приемника считывает записи данных из одного или нескольких разделов шины данных и записывает эти записи в одно из хранилищ данных.

Основной задачей соединителя приемника является надежная отправка данных в целевое хранилище данных, гарантируя, что данные не будут потеряны или дубликаты не появятся из-за перезапуска соединителя. Разъем приемника должен поддерживать положение уже загруженных данных и позволять сбросить это положение для повторной загрузки данных в случае необходимости.

DATASKAI не предоставляет готовых исходных соединителей, но вместо этого поддерживает использование сторонних реализаций соединителей приемников, предназначенных для различных типов хранилищ данных в хранилище данных.

Могут использоваться существующие общие коннекторы для Kafka (например, опубликованные на Confluent Hub) или разработать собственные коннекторы. Рекомендованным способом реализации коннектора приемника является использование API Java и Kafka Connect.

В случае невозможности, соединитель приемника может быть реализован с использованием любого языка программирования с доступной клиентской библиотекой Kafka.

7.2 Data Storage Subsystem

Подсистема хранения данных осуществляет управление и хранение данных, используемых платформой. Это включает в себя необработанные данные, собранные из внешних источников, и производные данные, полученные компонентами DATASKAI, например, функции. Подсистема также предоставляет удобный сервис для обнаружения и доступа к хранимым данным для компонентов DATASKAI и пользователей.

Хранимые данные могут быть разных типов (реляционные данные, временные ряды, файлы и т. Д.) и объемов и могут иметь разные схемы доступа. Чтобы эффективно удовлетворить такие разнообразные потребности, подсистема использует подход polyglot persistence, полагаясь на несколько внутренних хранилищ данных, оптимизированных для различных вариантов использования. Подсистема позволяет использовать различные реализации хранилища данных в качестве внешних зависимостей.

Подсистема скрывает сложные структуры базовых хранилищ данных от своих клиентов, предоставляя высокоуровневую службу данных для обнаружения и доступа к данным.

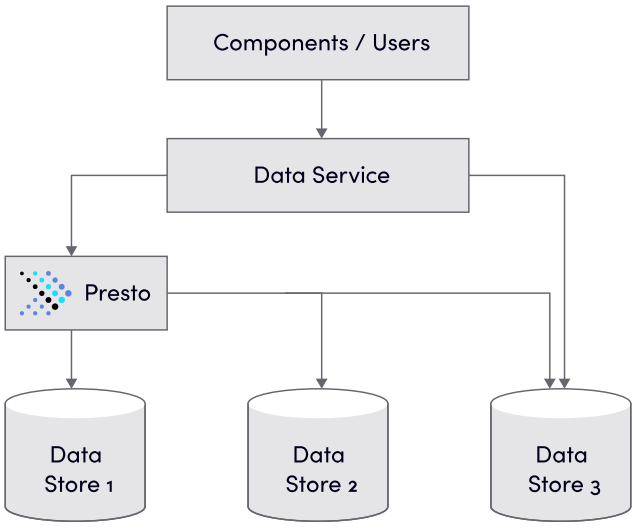

Рис. 3. Диаграмма Data Storage Subsystem

Хранилища данных

Используемые хранилища данных можно разделить на следующие типы:

– **Система управления реляционной базой данных Relational database** management system, которая подходит для хранения (полу) структурированных данных относительно небольшого объема, с возможностью эффективного доступа к данным с использованием SQL. Рекомендуемое решение - PostgreSQL.

– **Масштабируемая распределенная база данных или хранилище Scalable distributed database or warehouse**, которая подходит для хранения (полу) структурированных данных большого объема с возможностью эффективного доступа к данным с использованием SQL или другого языка запросов. Примеры: Cassandra, Elasticsearch, Hive.

– **База данных временных рядов Time series database**, оптимизированная для хранения, доступа и обработки данных временных рядов, например, измерения или события, собранные в разные моменты времени. Эти решения поддерживают эффективный доступ к таким данным, включая чтение по временным интервалам, агрегирование, понижающую дискретизацию и т. Д. Примеры: TimescaleDB, InfluxDB, OpenTSDB.

– **Файловое хранилище File storage**, которое предоставляет универсальное решение для хранения произвольных данных в форме файлов, включая неструктурированные данные. Распределенные файловые системы и хранилища объектов позволяют организовать надежное хранение и параллельную обработку больших наборов данных. Рекомендуемые решения - файловый сервер или NAS для небольших объемов данных и кластер HDFS для больших объемов данных.

В зависимости от предметной области набор используемых хранилищ данных может различаться. Минимальная конфигурация использует PostgreSQL для хранения всех данных в подсистеме.

7.2.1 Data Service

Служба данных реализует высокоуровневый интерфейс для обнаружения и доступа к данным, хранящимся в подсистеме. Это позволяет скрыть от клиентов сложность и неоднородность базовых хранилищ данных.

Она выполняет две функции:

1) обнаружение данных и управление метаданными. Сервис регистрирует наборы данных, то есть коллекции записей данных, хранящиеся в подсистеме. У каждого набора данных есть уникальный идентификатор, который можно использовать для ссылки на него. Наборы данных организованы в пространства имен. Служба данных хранит метаданные для каждого набора данных, включая его имя, описание, теги, схему и др. Он также хранит информацию, необходимую для доступа к набору данных – расположение данных во внутреннем хранилище данных. Интерфейс службы данных включает операции для запросов и управления наборами данных.

2) доступ к данным. Позволяет своим клиентам запрашивать и загружать данные из заданного набора данных в одном из поддерживаемых форматов (csv, tsv, json и другие). Data Service поддерживает гибкий язык запросов, который позволяет клиентам запрашивать произвольные подмножества сохраненных данных, задавая необходимые поля данных и фильтры. Служба данных также может реализовывать дополнительные операции, такие как сортировка данных, псевдонимы полей, преобразование значений полей и др. Служба данных реализует доступ только для чтения к сохраненным данным, операции записи выполняются путем прямого доступа к внутренним хранилищам данных.

Сервис реализован как REST-сервис на Java.

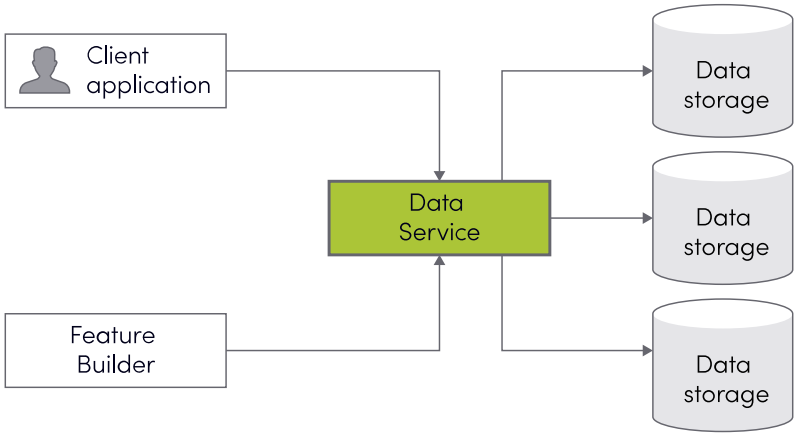

Рис. 4. Data Service

Сервис может использоваться как конечным пользователем, так и другими сервисами, например, для работы экстракторов признаков, получения предыдущих прогнозов данных и т. д.

7.3 Data Analysis Subsystem

Подсистема включает в себя инструменты, которые выполняют анализ данных, состояние объектов домена и их функции. Результатами анализа могут быть, например, мониторинг событий (предупреждений), аналитика (статистика поломок объектов) и прогнозы (поломки в будущем). Ниже перечислены основные компоненты подсистемы: служба моделей, служба пакетного запуска моделей, модуль мониторинга данных и аналитика / BI.

Model service

Этот компонент построен на архитектуре ”клиент-сервер”и предназначен для работы с моделями машинного обучения / алгоритмами принятия решений в качестве службы.

Для использования в сервисе модель оборачивается вспомогательным кодом - оберткой. Оболочка скрывает от автора модели сетевое взаимодействие, операции регистрации и операции по нахождению и управлению моделью, предоставляя лишь несколько абстрактных методов («запуск», «остановка», «прогноз»). Снаружи оболочка предоставляет интерфейс для удаленного вызова (gRPC), в том числе по сети.

Запрос к сервису модели выполняется по протоколу HTTP (GET, POST). Оригинальная версия сервиса поддерживает запросы на запуск и остановку моделей, получение прогнозов, загрузку и скачивание архивов и другие (см. Player Models Player).

Сервис моделей пакетного обслуживания Models batch run service

Предназначен для автоматизации запуска моделей с заданным набором функций и сохранения результатов в подсистеме хранения данных.

Типичным примером использования этой услуги является периодическая генерация прогнозов для постепенно обновляемых данных с использованием набора моделей. Компонент реализован как сервис REST в Java.

Алгоритм отправки запроса в сервис состоит из следующих шагов:

– загрузка требуемых значений из хранилища объектов (например, по номеру сборки объектов, временному интервалу, объектам);

– получение списка необходимых моделей с их настройкой через сервис моделей;

– подготовка запросов на модели (особенности векторов);

– вызовы нужных моделей через сервис моделей, сбор полученных результатов;

– сохранение результатов в подсистеме хранения данных (СУБД).

Мониторинг данных Data monitoring

Назначение этого модуля - управлять параметрами объектов в предметной области и генерировать события в соответствии с заданными правилами. Входные данные - это особенности объектов домена в дискретных временных выборках, предоставляемых подсистемой хранения. Выходные данные представляют собой графическое представление временных рядов атрибутов объектов домена и связанных событий. Модуль реализован на основе открытого продукта Grafana, разработан адаптер и используется для подключения к подсистеме хранения.

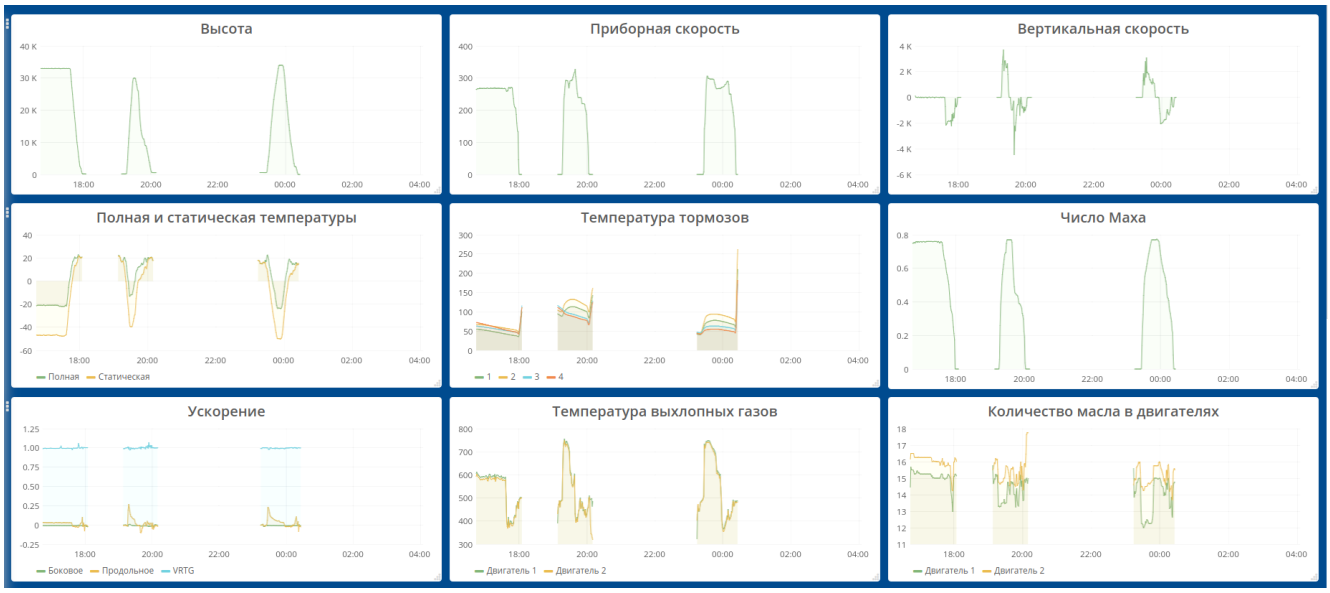

Рис. 5. Пример интерфейса для мониторинга данных

Аналитика / BI

Модуль позволяет генерировать пользовательские аналитические отчеты на основе данных, содержащихся в подсистеме хранения, создавать отчеты о распределении, отображать важные показатели в виде веб-страниц.

Модуль переводит запросы пользователей в запросы к подсистеме хранения, фильтрует и объединяет данные для выпуска окончательных отчетов. Для реализации используются open source Metabase и Apache Superset.

7.4 Data Processing Subsystem

Подсистема реализует масштабируемую вычислительную инфраструктуру для выполнения обработки данных по запросу компонентов DATASKAI. Также на этом уровне пользователи должны описать предметную область - абстракцию предметной области для вышележащих слоев. Данные представляются в виде объектов с привязкой данных.

Вычислительный кластер Computing cluster

Вычислительная инфраструктура основана на кластере, состоящем из нескольких серверов и поддерживающем горизонтальное масштабирование за счет увеличения количества серверов.

Кластер должен быть подключен к высокоскоростной сети с подсистемой хранения данных. Часть хранилища (например, распределенная файловая система) может быть размещена непосредственно на серверах кластера для оптимизации доступа к данным.

Для управления ресурсами кластера и выполнения заданий мы используем существующие решения в форме Hadoop YARN и Kubernetes.

Модуль управления процессом Process control module

Цель этого модуля - регистрировать, запускать и контролировать выполнение заданий обработки данных, состоящих из нескольких задач с зависимостями между ними. Примером является процесс обновления прогнозов, который включает создание набора объектов для объектов и вызов моделей с векторами, состоящими из значений объектов. Модуль должен автоматизировать выполнение таких процессов, включая запуск отдельных задач и обработку сбоев.

В общем случае описание процесса представляет собой ориентированный ациклический граф, вершинами которого являются задачи (запуск указанного модуля с указанными входными данными), а ребра определяют зависимости данных (для задачи А нужны данные из задачи В) или управление (задача А можно запустить только после выполнения задачи Б).

В задачи модуля управления процессом входит:

– хранение описаний процессов, загруженных пользователями;

– выполнение процессов по запросу или по графику;

– запуск задачи в соответствии с их зависимостями от вычислительной инфраструктуры через менеджер ресурсов;

– мониторинг задач и обработка сбоев;

– предоставление информации о состоянии процессов.

В настоящее время модуль реализован как сервис на Java и предоставляет REST API для инициализации запуска предварительно сконфигурированных процессов, например построения прогнозов. Также можно настроить автоматический запуск процесса с помощью выражений cron. Кроме того, этот сервис действует как единая точка входа, позволяя выполнять дополнительные функции перед отправкой запроса.

Пример запроса на запуск конвейера:

```
POST/api/pipeline_runner/predict?From =
1504224000 & amp; to = 1504310400 HTTP / 1.1
Host: 172.17.0.1:8480
cache-control: no-cache
```

Пример описания сервиса:

```
http://localhost:8480/swagger-ui.html # /
```

Во многих случаях обработка данных в DATASKAI может быть тривиально распараллелена данными, например, при построении объектов для отдельных объектов. Для таких случаев необходимо создать и запустить необходимый набор независимых задач.

В более сложных случаях, например, для аналитики всего набора данных, планируется использовать существующие решения, такие как Apache Spark, для организации распределенной обработки данных в кластере.

Модели доменов Domain models

Разработаны в методологии объектно-ориентированного программирования на языке высокого уровня (предоставляется поддержка языка Python). Предметная область описывается в форме классов, которые имеют разные связи между ними. Объекты, созданные из классов, связаны с данными в соответствии с механизмами, описанными в конструкторах объектов. Модели разрабатываются в соответствии с процессами заказчика, на основе которых предполагается формулировать задачи для работы аналитической команды.

Реализация модели может быть выполнена как с инициализацией доступа к данным непосредственно в конструкторе, так и с использованием «фабричного» шаблона программирования, когда другой объект, фабрика, используется для построения объекта домена. В этом случае построение объекта домена выполняется путем вызова метода фабрики.

Для доступа к уровню данных при создании моделей используются компоненты доступа к данным (API и клиенты подсистемы хранения данных).

Использование шаблона «фабрика» является предпочтительным, посколь-

ку его можно использовать в качестве объекта, который хранит соединения с системами доступа к данным и промежуточный кэш уже созданных объектов. Таким образом, можно избежать потери времени на повторную инициализацию соединений и строительных объектов.

Модуль экстракторов Feature extractors module

Основное назначение модуля – обработка последовательностей объектов во временных рядах признаков. Модуль выполняет построение функций в несколько этапов:

– построение основных функций: необходимые функции извлекаются из объекта домена, настроенного в качестве цели в постановке задачи, а также из связанных объектов, и они денормализуются;

– векторизация денормализованных символов, полученные векторы признаков выкладываются в хранилище признаков;

– построение вторичных объектов: векторы объектов запрашиваются из хранилища объектов, и по их выбору запускаются конструкторы объектов, которые учитывают временную зависимость между соседними векторами; функции, полученные на предыдущем этапе, выкладываются в хранилище функций.

Разработка процессов извлечения первичных признаков осуществляется на языке высокого уровня, совместимом с языком описания объекта предметной области. Процессы извлечения вторичных признаков могут быть разработаны на любых языках, совместимых с DATASKAI.

В распоряжении пользователей имеется библиотека для извлечения вторичных признаков. Возможно исполнение композиций признаков. Построение вторичных признаков выполняется на временных рядах первичных и вторичных признаков, в то время как инструменты для извлечения вторичных признаков могут корректировать свои внутренние коэффициенты в соответствии с характеристиками обучающего набора.

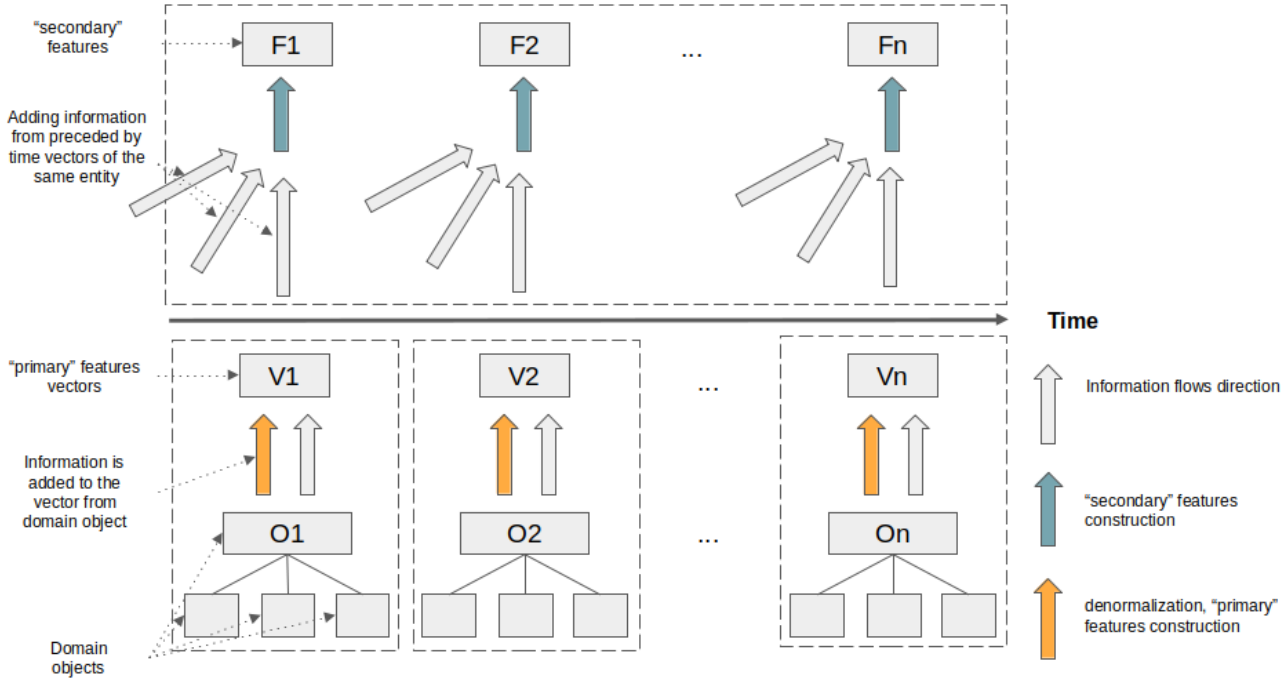

Рис. 6. Схема получения признаков

7.5 User tools and extensions

Обеспечивают взаимодействие администраторов и конечных пользователей (аналитиков, отраслевых экспертов, менеджеров) с DATASKAI, включая визуализацию данных через графический интерфейс, аналитику и вывод прогноза, чтобы помочь в принятии решения задач.

Data Science Toolkit

Реализует возможность решения типичных задач по науке о данных с помощью инструментов DATASKAI Python. Может использоваться таких задач, как:

– загрузка функций и необработанных данных из соответствующих хранилищ;

– загрузка целевого вектора;

– представление моделей и результатов моделирования;

– работа с компонентом Models Player из Python;

– проведение специальных перекрестных проверок данных временных рядов.

В настоящее время Data Science Toolkit представлен на платформе в качестве компонента Evaluation Tools.

Компоненты оценки моделей Models evaluation components

DATASKAI обеспечивает автоматическую оценку обученных ML-моделей или алгоритмов принятия решений (далее именуемых моделями) и интерфейс для просмотра текущих метрик моделей, нацеленных на оценку.

Эта функциональность реализована двумя сервисами. Первый, Metric Service, выполняет расчет метрик. Он предусматривает добавление внешних метрик для оценки моделей посредством конфигурации постановки задачи.

Расчет метрик выполняется на статической выборке, указанной в качестве теста при постановке задачи. Модуль автоматически выполняет обновление метрик в случае добавления новых, изменения тестового образца, добавления новых моделей для оценки.

Второй, Submits Web App, отвечает за визуализацию результатов. Веб-приложение отравки сабмитов возвращает таблицу, содержащую список моделей, отправленных для оценки, с их именами, авторами, значениями ошибок по указанным метрикам, тегам и т. Д. Таблица поддерживает возможность сортировки и фильтрации результатов моделирования.

Интерфейсы администратора Admin Interfaces

Визуальный мониторинг осуществляется через подсистему мониторинга, а компоненты программного комплекса управляются через интерфейсы администратора. Запуск, остановка и управление компонентами выполняются с помощью автоматизированных инструментов Kubernetes, Docker Compose. Регистрация сервисов осуществляется через механизмы централизованного сбора журналов Graylog непосредственно из запущенных контейнеров приложений.

Пользовательские интерфейсы и внешние приложения и клиенты

Визуализация собранных данных и результатов анализа выполняется через внешний графический интерфейс пользователя, который получает данные через API, который работает поверх подсистемы хранения. Внешние приложения и клиенты ориентированы на конкретные области приложений DATASKAI и могут создаваться сторонними разработчиками путем интеграции с его открытыми API.

7.6 Service subsystems

Мониторинг Monitoring

Эта подсистема контролирует состояние компонентов DATASKAI. Каждый компонент публикует пользовательский набор метрик в подсистеме мониторинга. Подсистема собирает и хранит значения метрик, предоставляя клиентам (например, интерфейсам администратора) доступ к текущим значениям и истории. Подсистема также генерирует уведомления в соответствии с пользовательскими правилами.

Для реализации подсистемы мониторинга используются открытые решения Graphite и Grafana. Графит выступает в качестве хранилища метрик, Grafana реализует настраиваемые графические интерфейсы для мониторинга метрик онлайн. Подсистема мониторинга используется для быстрого поиска «узких мест» в программном пакете, а также для быстрой отладки.

Безопасность Security

Безопасность данных достигается путем изоляции компонентов внутри защищенных сетей, оставляя только шлюз для отправки данных извне.

Встроенная подсистема безопасности реализует учет пользователей программного пакета, их аутентификацию и авторизацию. Доступ пользователей к интерфейсам DATASKAI настраивается на уровне ролей пользователей, таких как администратор, ученый данных, целевой пользователь.

8 Компоненты

8.1 Evaluation tools

Инструменты оценки обеспечивают инструментарий data science.

Описание высокого уровня

Компонент реализует следующие функции:

– загрузка функций и необработанных данных из соответствующих хранилищ;

– загрузка целевого вектора;

– представление моделей и результатов моделирования;

– работа с компонентом Models Player из Python;

– проведение специальных перекрестных проверок данных временных рядов.

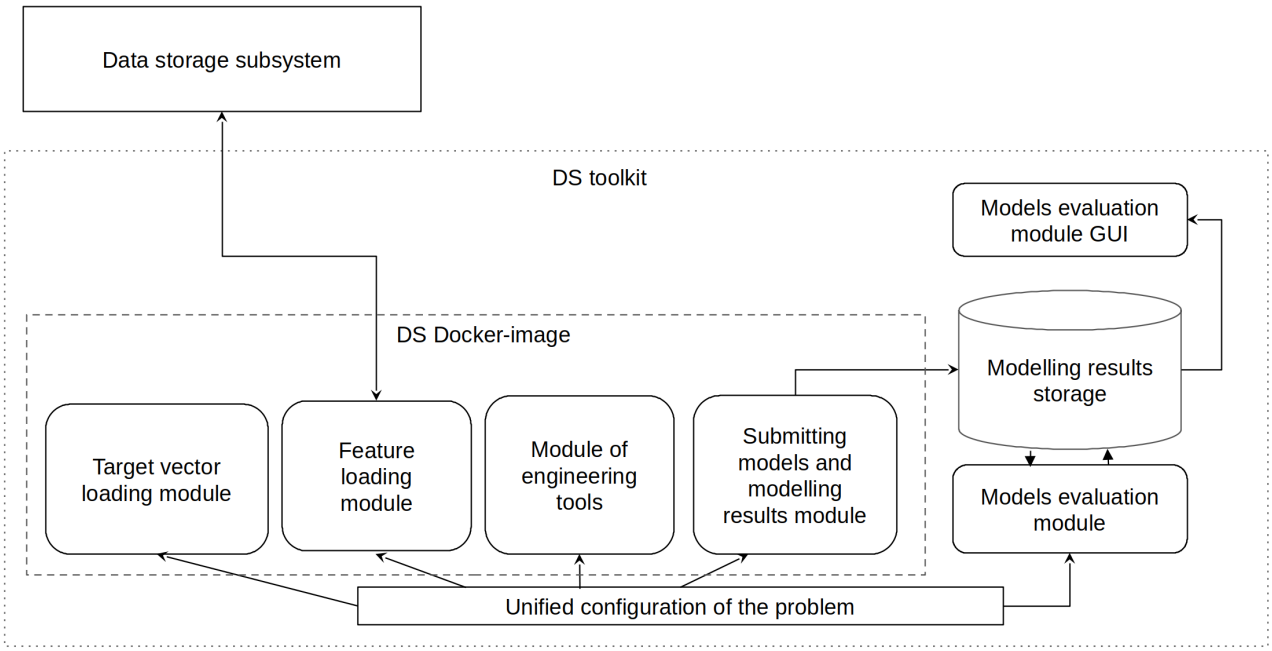

Рис. 7. Инструменты для обработки данных

Инструментарий построен таким образом, чтобы обеспечить возможность вертикального и горизонтального масштабирования работы над задачей. Конфигурация проблемы - это описание модели предметной области, образцы объектов предметной области, целевая переменная, тип проблемы, метрики для оценки результатов моделирования.

Для работы с инструментарием аналитикам предоставляется доступ к докконтейнеру, в котором все необходимые библиотеки предустановлены для выполнения интеллектуального анализа, визуализации, работы с различными источниками данных, построения моделей машинного обучения и каркасов для работы с нейронными сетями. Изображение этого контейнера используется не только для работы аналитиков, но и для воспроизведения моделей, сборки документации и построения функций.

Все эти процессы чрезвычайно чувствительны к управлению версиями их компонентов. Различия в версиях библиотек в работе аналитиков и в производстве приводят к ошибкам в предсказаниях моделей, различиям в построенных функциях. Docker решает эту проблему, и обновление всей инфраструктуры сводится к обновлению базового образа. В то же время это решение является безопасным, поскольку аналитик работает в изолированной среде и не имеет доступа к команде sudo и, следовательно, к содержимому сервера, на котором развернут контейнер с инструментарием ученого по данным. Наличие образа также создает возможность быстрого развертывания и свертывания работы групп аналитиков, масштабируя их на количество аппаратных / инженерных задач и, среди прочего, облегчает переход текущей среды на новые вычислительные мощности.

Все инструменты в компоненте представлены в виде объектов Python. Рекомендуется начать работу с инициализации экземпляра класса TaskLoader. Инициализация выполняется с указанием конкретной задачи машинного обучения.

После создания TaskLoader получает атрибуты **raw_data_loader, feature_loader, target_loader** и **submitter**, которые предназначены, соответственно, для загрузки необработанных данных, загрузки объектов, загрузки целевой переменной и отправки результатов моделирования в подсистему оценки. raw_data_loader, feature_loader, target_loader и submitter по сути являются объектами классов
RawDataLoader, ‘FeatureLoader‘, TargetLoader и Submitter, которые также можно использовать (но не рекомендуется) без использования TaskLoader.

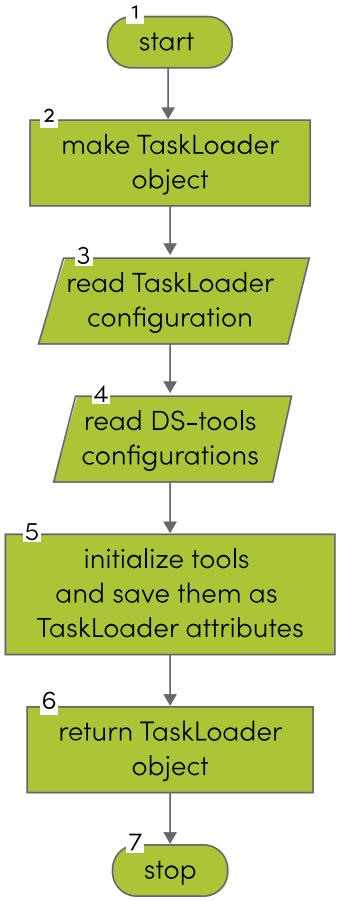

Рис. 8. Блок-схема работы загрузчика задач

RawDataLoader предназначен для загрузки необработанных данных для проведения исследовательского анализа данных. Теперь данные могут быть загружены только по сети из службы данных. Вам нужно указать record_to_use с именем из конфигурационного файла и опционально функцией выбора столбцов. Также возможно скачивать только типы данных.

FeatureLoader выполняет загрузку функций для обучающей и тестовой выборки из хранилища данных с учетом постановки задачи. Модуль обеспечивает настройку параметров для выгрузки объектов путем определения функции для их выбора. Также возможно загрузить типы данных и список доступных конфигураций, каждая из которых описывает определенный набор функций. Данные могут быть загружены по сети или из локальных файлов.

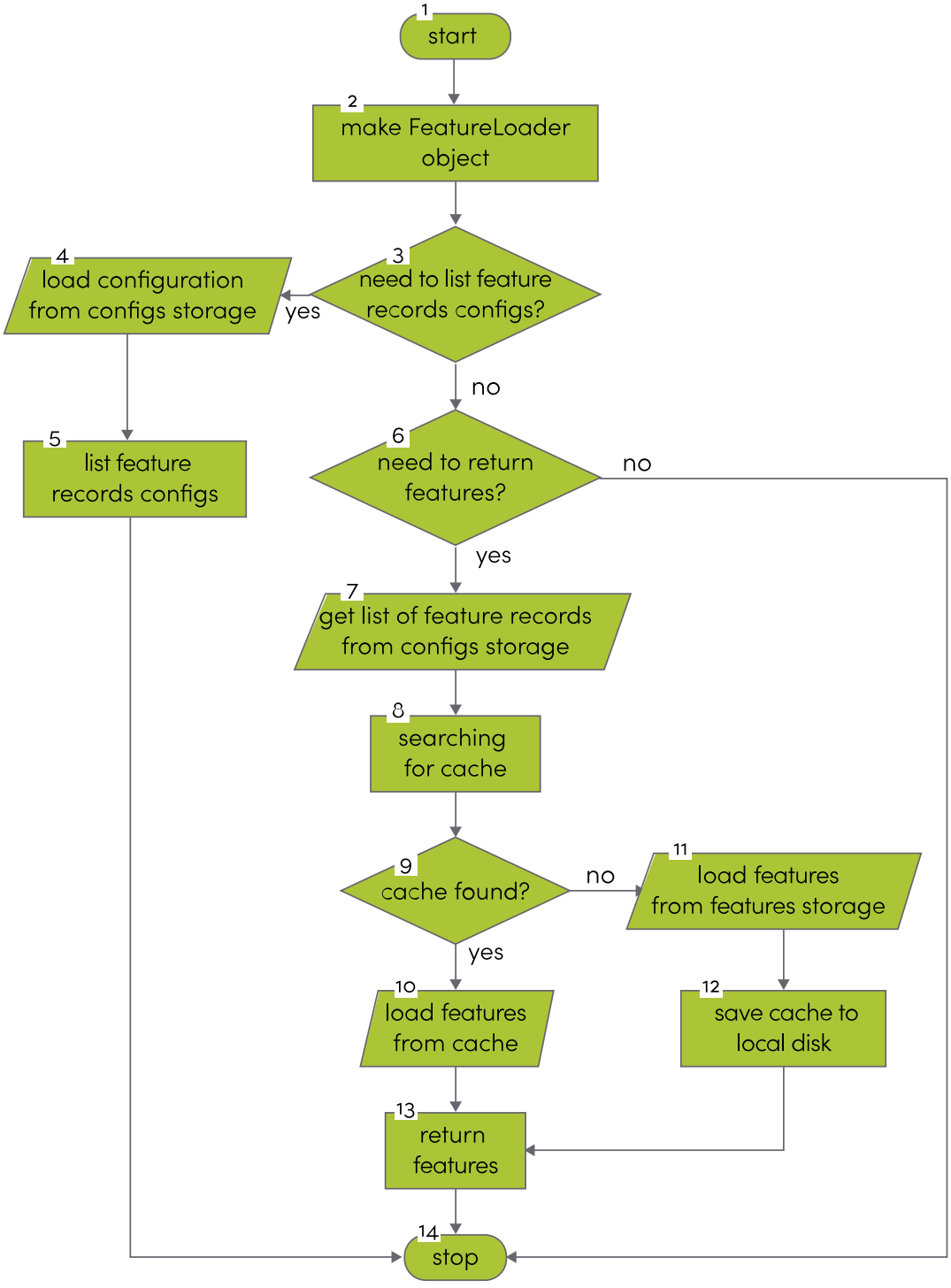

Рис. 9. Блок-схема работы Feature Loader

TargetLoader обеспечивает доступ к целевому вектору для обучающей выборки в соответствии с формулировкой задачи. Он также имеет возможность заранее при настройке модуля исправить метод разделения образца для проверки.

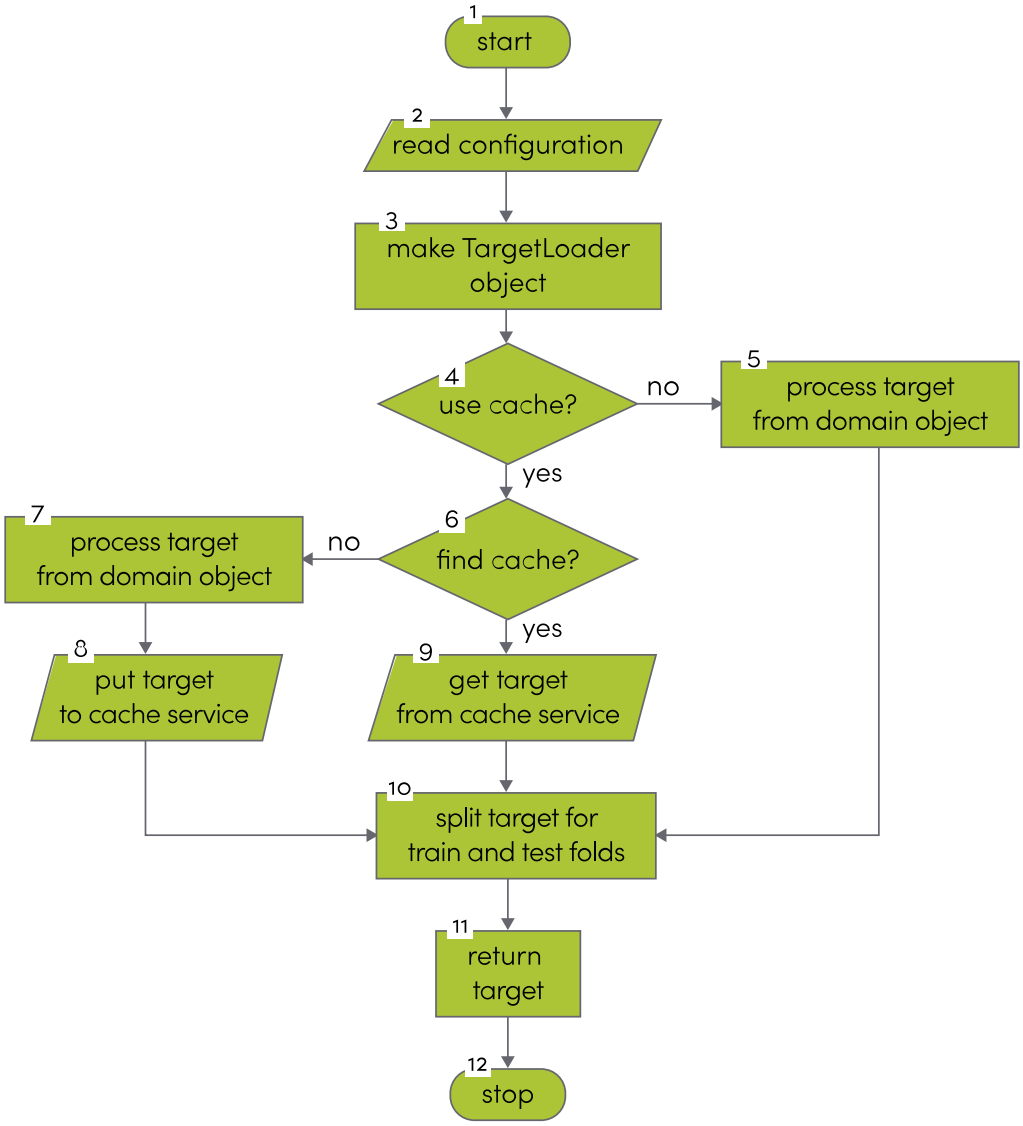

Рис. 10. Блок-схема работы Target Loader

Submitter обеспечивает проверку и отправку моделей и результатов моделирования в подсистему оценки моделей. Во время процедуры проверки проверяется соответствие представленных данных постановке проблемы, а именно:

– имя отправленного целевого вектора для расчета метрик;

– размер целевого вектора;

– наличие модельного объекта;

– вектор значимости признаков (для определенных типов задач);

– объект, который выполняет масштабирование признаков перед применением алгоритма / модели;

– значения по умолчанию для заполнения пробелов во входных векторах.

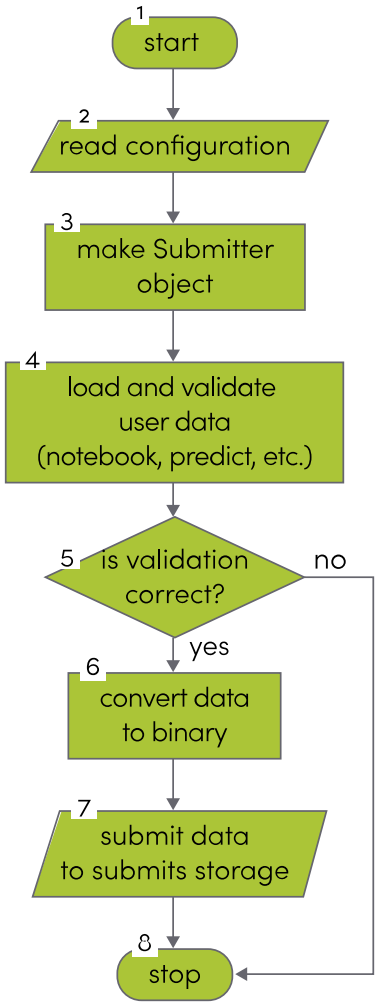

Рис. 11. Блок-схема работы отправителя

Метаинформация также вводится в подсистему оценки, что позволяет осуществлять быстрый поиск в хранилище результатов моделирования. Набор результатов моделирования данных, отправляемых в хранилище, включает в себя:

– результат алгоритма / модели принятия решения на тестовом образце;

– Python-объект алгоритма / модели принятия решения, дополнительная информация (список и важность функций, способ работы с пропущенными значениями и т. Д.);

– jupyter-блокнот с описанием решения проблемы и конструкцией модели;

– имя автора, название и версия модели, связанные теги и комментарии.

Module of engineering tools содержит вспомогательные инструменты для работы над науками о проблемах в своих высказываниях. Если возможно, инструменты реализуются с учетом интерфейса распространенной ныне склеарной библиотеки Python.

В частности, в соответствии с интерфейсами sklearn были реализованы специальные инструменты для работы над задачами, ориентированными на изучение временных рядов: классы Python для разделения наборов данных на складки для выполнения различных типов перекрестной проверки.

Модуль поддерживает следующие функции:

– получение складок из временных рядов по нескольким схемам с учетом специфики распределения причинно-следственных связей во времени; – подготовка данных в несбалансированных выборках, содержащих временные ряды: данные делятся на сгибы с учетом принадлежности к конкретным объектам, так что в результате линии, относящиеся к одному объекту, равномерно распределяются по всем сгибам, другими словами, что каждая подвыборка содержит максимальное количество строк, связанных с различными объектами; – получение сгибов с фокусом на значениях в целевом векторе: в качестве основы для сгибов выбираются группы смежных рядов, для которых целевая переменная принимает определенное значение (один); устанавливается «окно» сэмпла (количество строк, непосредственно предшествующих каждой группе по времени и непосредственно следующих за ней). Тогда линии, которые не входят ни в одну из складок, равномерно распределяются между ними. Алгоритм обеспечивает наличие строк с определенным значением целевой переменной в каждом из сгибов.

Кроме того, DATASKAI реализует несколько схем выбора признаков методом обертки, используя набор деревьев решений, а также генетический алгоритм, который позволяет осуществлять поиск путем кодирования пространства в виде двоичного вектора и последующей оптимизации. Генетический алгоритм также может быть использован для поиска оптимальных гиперпараметров моделей.

В сочетании с инструментом для загрузки функций эти модули создают предпосылки для реализации AutoML-схем, другими словами, автоматического создания алгоритма / модели принятия решений на основе входных данных.

8.2 Toolkit CLI

Инструменты для облегчения обработки данных с помощью DATASKAI.

Описание высокого уровня

Все инструменты представляют собой скрипты Python для командной строки, которые могут упростить операции обработки данных.

В настоящее время представлено несколько инструментов:

get_extractor_info.py, make_config.py, make_dataset.py, send_extractor.py.

get_extractor_info.py - показывает список имен или конфигурационных файлов существующих экстракторов.

send_extractor.py - отправка экстракторов объектов в сервис Feature Builder.

make_config.py - создание конфигурации для отправки данных в службу данных.

make_dataset.py - создание / загрузка набора данных в уже существующий набор данных.

8.3 Submits Web App

Отображает графический интерфейс лидеров для представлений

Описание высокого уровня

Submits Web App - это веб-сервис, который используется для отображения результатов моделирования с помощью графического интерфейса пользователя.

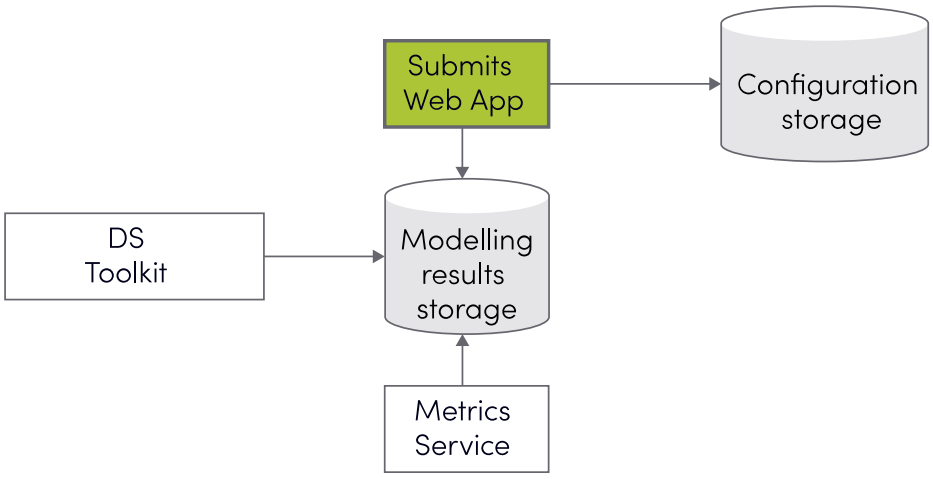

Рис. 12. Диаграмма взаимосвязей Submits Web App

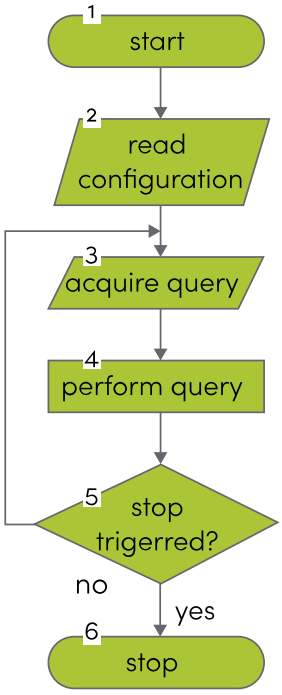

Рис. 13. Блок-схема

Алгоритм компонента максимально прямолинейный: фактически представляет собой чтение настроенных отправляемых коллекций и их визуализацию в виде таблицы.

Описание графического интерфейса

После запуска сервис предоставляет веб-интерфейс на 5000 портов (по умолчанию). Интерфейс состоит из основной таблицы с полями (указанными в конфигурации сервиса) и элементами управления.

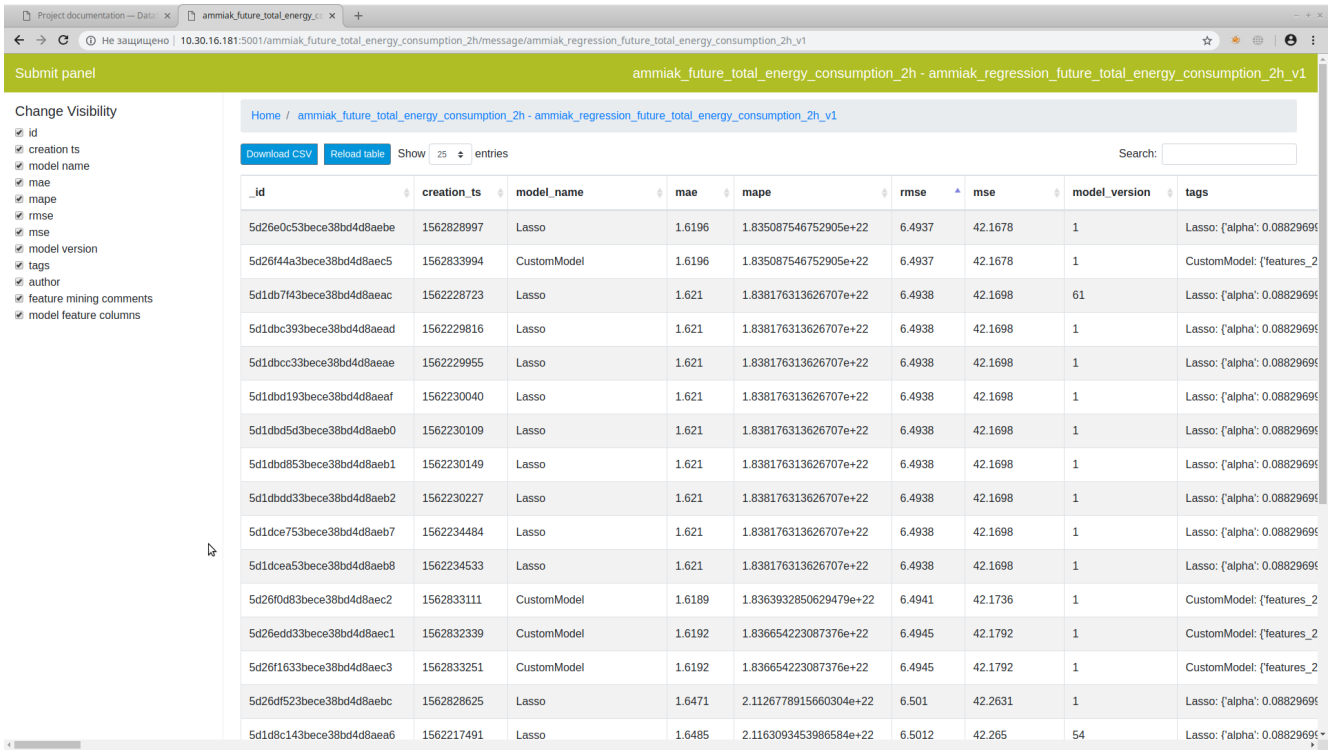

Рис. 14. Веб-интерфейс Submits Web App

8.4 Metrics Service

Описание высокого уровня

metrics_service - это сервис, который используется для расчета метрик, настроенных для задачи. Расчет метрик для каждой задачи выполняется с помощью соответствующего metrics_service, поэтому он должен быть настроен соответствующим образом для каждой задачи.

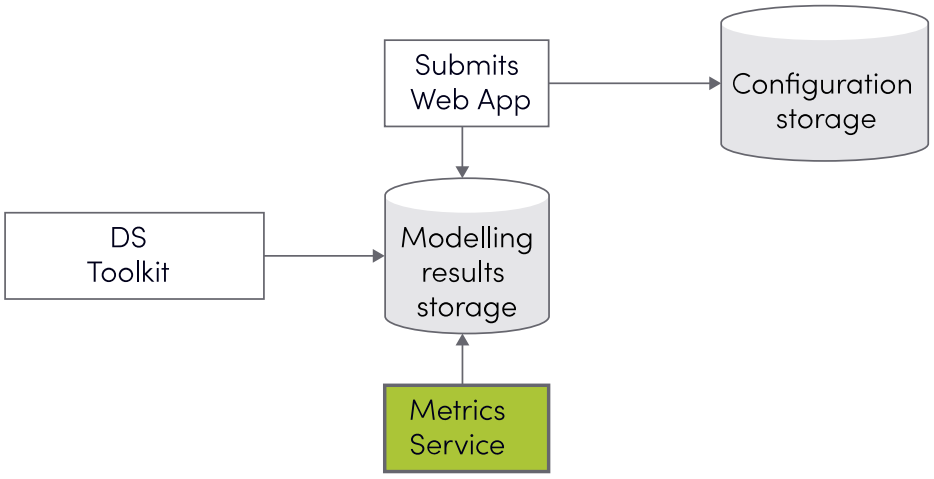

Рис. 15. Взаимодействие Metric Service с другими компонентами и хранилищами данных

Алгоритм metrics_service загружает конфигурацию и выполняет расчеты метрик для представленных результатов. Он также автоматически перезагружает конфигурацию без перезапуска.

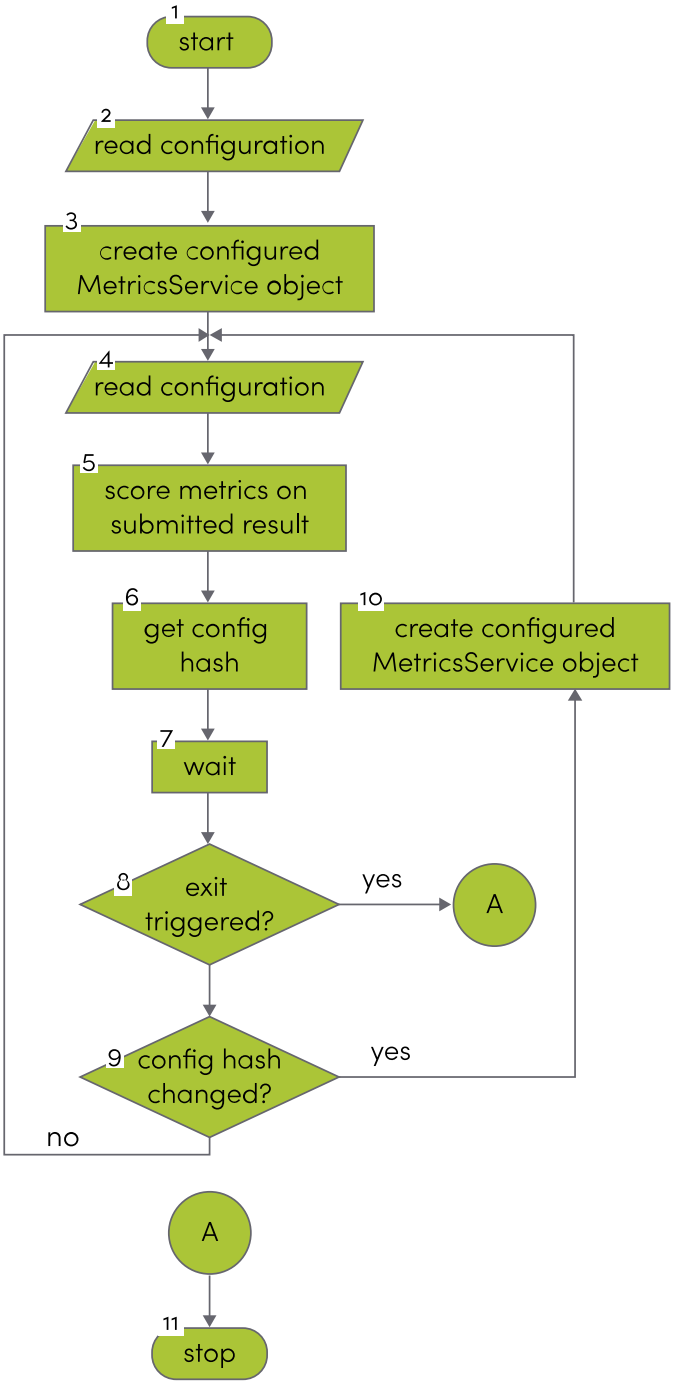

Рис. 16. Схема процесса

Описание сервиса

После запуска служба пытается подключиться к хранилищу конфигурации и отправляет хранилище. Когда соединение установлено, сервис начинает работать. Работа metrics_service представляет собой бесконечный цикл двух основных действий:

– проверка текущих изменений конфигурации;

– расчет метрик по заданному заданию.

Если конфигурация изменяется, сервис перезагружает настройки и продолжает работать с новой конфигурацией.

Процесс расчета метрик выделен в отдельный сервис по следующим причинам:

– при работе с реальным бизнес-клиентом метрики изменяются. Практически в каждом проекте разработчики оказывались в ситуации, когда заказчик просил рассчитать «еще одну метрику», которая не была предложена в начале проекта.

– при необходимости адаптировать процесс проверки метрик не только к потребностям клиентов, но и к данным. Это может быть недействительно через некоторое время, поэтому требуется пересчитать все, кроме этих регионов.

– есть необходимость проверять некоторые другие показатели самостоятельно. Требуется добавить наши собственные метрики.

– есть необходимость подводить итоги некоторых моделей и получать из них лучший ансамбль.

Если бы метрики редактировались из ноутбуков, существовала бы необходимость перезапускать каждый из ноутбуков, связанных с работой DS. Такие пересчеты приводят к бессмысленной трате ресурсов и, более того, к возникновению дополнительных ошибок в коде ноутбуков.

Когда метрики отделены от прогнозов и записных книжек, такие пересчеты выполняются быстро, не беспокоя людей и не изменяя код записных книжек.

8.5 Project Wizard

Мастер развертывания программного комплекса.

8.6 Prediction Builder

Автоматизирует процесс вывода через REST API.

Описание высокого уровня

Сервис предоставляет возможность создавать, хранить и экспортировать прогнозы на основе функций из Feature Store и моделей из Models Player.

Поддерживает построение всех доступных прогнозов или прогнозов для конкретного объекта (объекта) экземпляра или модели.

Сервис зависит от Feature Store и Models Player и используется сервисом Pipeline Runner.

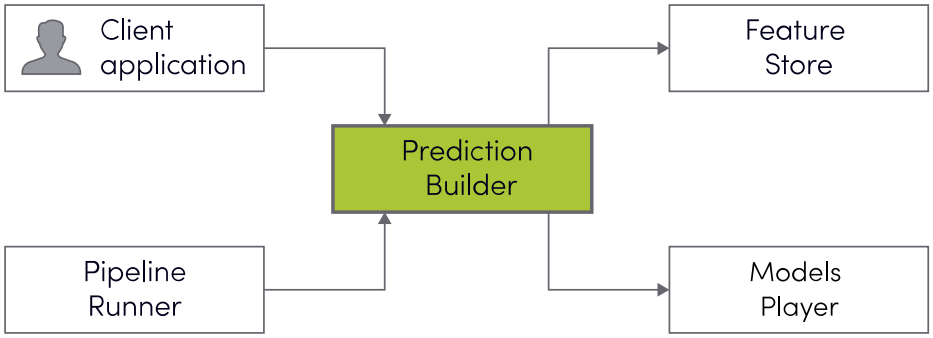

Рис. 17. Служба построения прогнозов взаимодействует с хранилищем функций и моделирует службы проигрывателя и используется исполнителем конвейера

Описание сервиса

API описывается с помощью Swagger, который автоматически генерирует сервисный веб-клиент. Он доступен по следующему адресу:

```
http://<hostname>:<port>/swagger-ui.html#/
```

8.7 Feature Store

Предоставляет REST API для работы с функциями.

Описание высокого уровня

Feature-store – сервис, отвечающий за предоставление доступа к функциям, вычисленным из наборов данных с помощью подключаемых экстракторов функций.

Функциональность сервиса включает в себя импорт функций из внешних источников в хранилище функций, возврат информации о сборках и чистых сборках.

Служба используется сервисами prediction_builder и feature_builder.

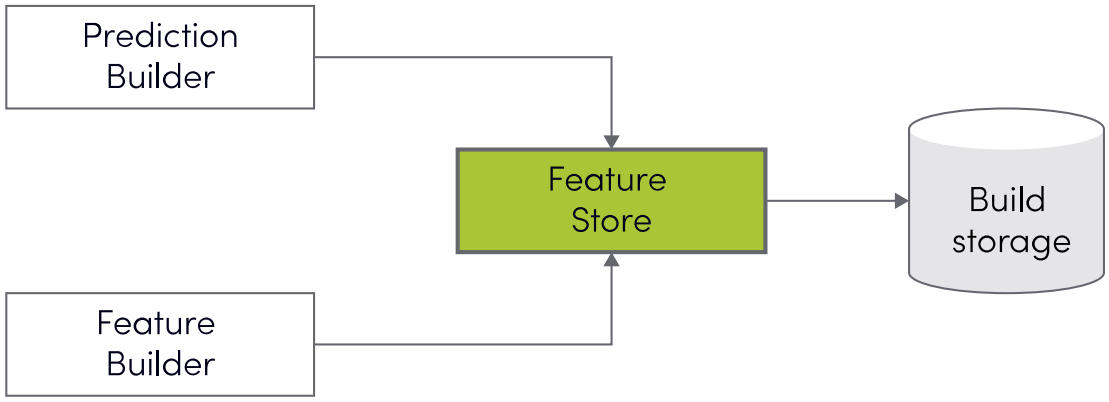

Рис. 18. Хранилище компонентов взаимодействует с хранилищем сборки и используется службами построения компонентов и прогнозирования

Описание сервиса

API описывается с помощью Swagger, который автоматически генерирует сервисный веб-клиент. Он доступен по следующему адресу:

```
http://<hostname>:<port>/swagger-ui.html#/
```

8.8 Models Player

Автоматизирует жизненный цикл процесса машинного обучения.

Описание высокого уровня

Models Player - веб-сервис Flask. Он работает как контроллер моделей в производственной среде. Он предоставляет функциональные возможности для запуска и остановки моделей, прогнозирования, получения рабочей статистики и т. д.

Внешний API для Player Models реализован в протоколе http. Внутренний API, с помощью которого Models Player работает и управляет моделями, реализованными в gRPC.

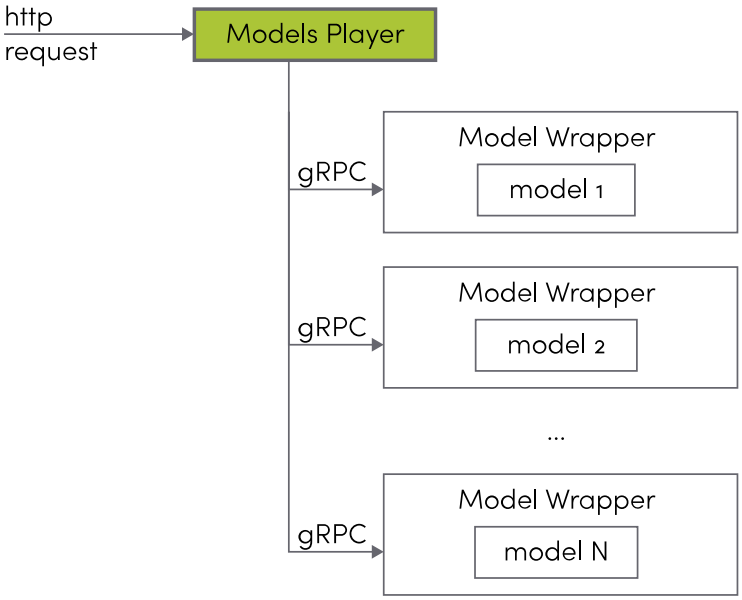

Рис. 19. Использование моделей упаковки в Model Player

Описание применения

Блок-схема компонента представлена на следующем рисунке.

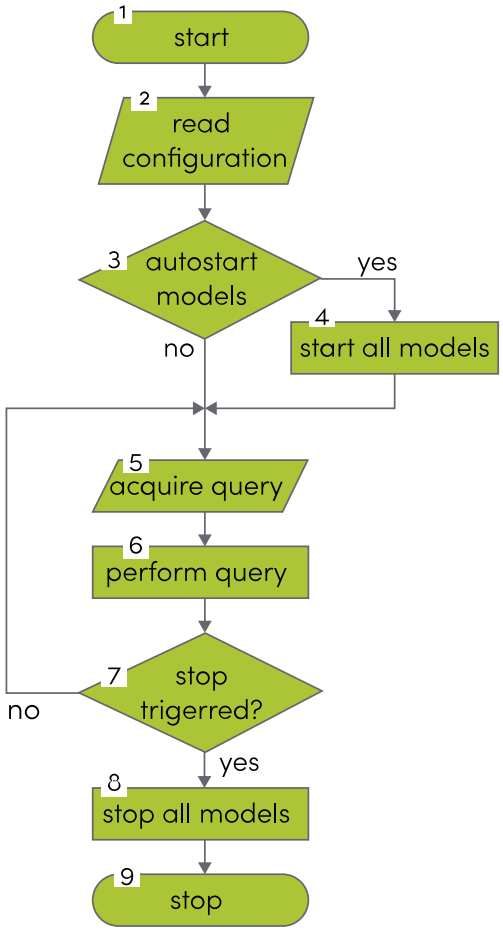

Рис. 20. Диаграмма Model Player

8.9 Model Wrapper

Упаковывает модель машинного обучения для совместимости с Models Player.

Описание высокого уровня

Оболочка работает как интерфейс абстрактной модели для связи со службой Models Player.

В DATASKAI модели машинного обучения работают как отдельные службы gRPC. Он берет на себя всю работу, связанную с проблемами масштабирования, используя микросервисный подход. Такой подход позволяет, например, запускать модели в независимых средах, писать их на разных языках и одновременно использовать один проигрыватель моделей.

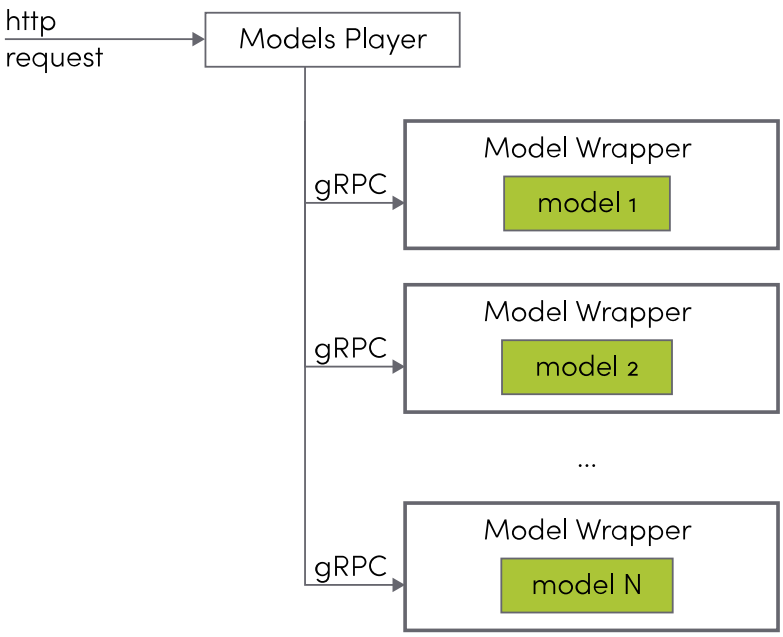

Рис. 21. Обёртывание модели для работы с Model Player

8.10 Feature Builder

Автоматизирует функции сборки процесса через REST API.

Описание высокого уровня

Feature Builder – служба, отвечающая за создание, хранение и предоставление доступа к функциям, вычисленным из наборов данных с помощью подключаемых экстракторов функций.

В случае успешного завершения метод / api / feature_builder / build создает BuildInfoObject с информацией о сборке объектов: количество объектов, для которых были построены объекты, идентификатор объектов и т. Д. Также метод возвращает параметр buildId, с помощью которого это возможно. сохранить функции в формате TSV.

Дополнительный функционал сервиса включает в себя возврат информации о сборках журнала.

Служба зависит от службы данных и хранилища компонентов и используется службами, выполняющими конвейер. Также он запускает экстракторы Python для запуска процесса сборки.

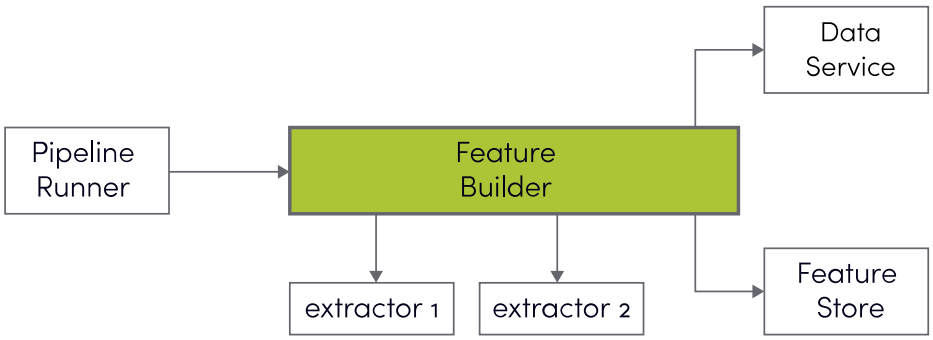

Рис. 22. Построитель признаков объектов взаимодействует со службой данных и хранилищем объектов и используется исполнителем конвейера

API описывается с помощью Swagger, который автоматически генерирует сервисный веб-клиент. Он доступен по следующему адресу:

```
http://<hostname>:<port>/swagger-ui.html#/
```

8.11 Data Service

Описание высокого уровня

Предоставляет REST API для запросов к хранилищу данных.

Выполняет две функции:

1) обнаружение данных и управление метаданными. Он регистрирует наборы данных, то есть коллекции записей данных, хранящихся в подсистеме. Каждый набор данных имеет уникальный идентификатор, который можно использовать для ссылки на него. Наборы данных организованы в пространства имен. Служба данных хранит метаданные для каждого набора данных, включая его имя, описание, теги, схему и т. Д. Она также хранит информацию, необходимую для доступа к набору данных, то есть местоположение данных во внутреннем хранилище данных. Интерфейс службы данных включает в себя операции для запроса и управления наборами данных;

2) доступ к данным. Он позволяет своим клиентам запрашивать и загружать данные из заданного набора данных в одном из поддерживаемых форматов (CSV, TSV, JSON и другие). Служба данных поддерживает гибкий язык запросов, который позволяет клиентам запрашивать произвольные подмножества хранимых данных, указывая необходимые поля данных и фильтры. Служба данных также может реализовывать дополнительные операции, такие как сортировка данных, наложение полей, преобразование значений полей и т. Д. Служба данных реализует только доступ только для чтения к сохраненным данным, операции записи выполняются путем прямого доступа к внутренним хранилищам данных.

Реализован в виде сервиса REST с помощью Java.

Сервис может использоваться как конечным пользователем, так и другими службами, например, для работы экстракторов признаков, получения предыдущих прогнозов данных и т. д.

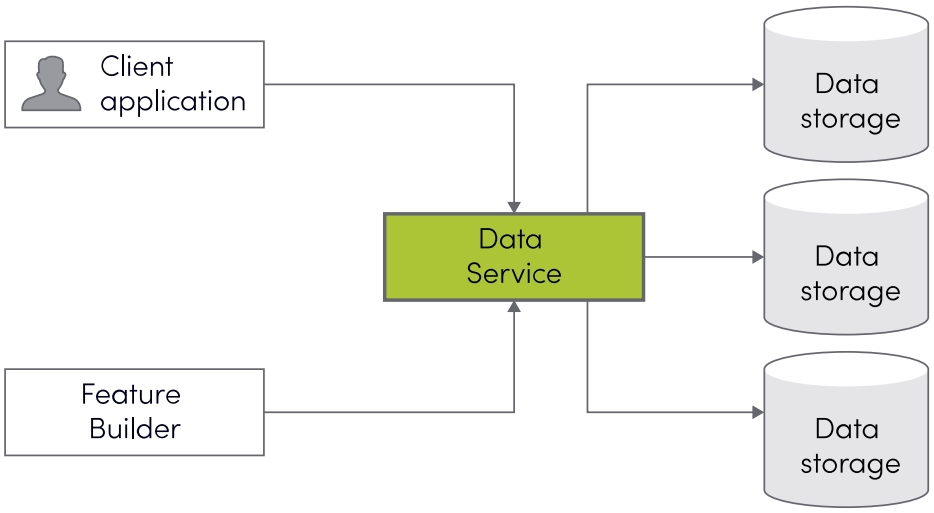

Рис. 23. Служба данных взаимодействует с хранилищем данных и используется службой построителя объектов.

Описание сервиса

API описывается с помощью Swagger, который автоматически генерирует сервисный веб-клиент. Он доступен по следующему адресу:

```
http://<hostname>:<port>/swagger-ui.html#/
```

8.12 Subject Domain Cache Service

Предоставляет возможность кэшировать различные пары типа ”ключ-значение”в базе данных Redis с помощью REST API.

Описание высокого уровня

Кэш предметного домена – это веб-сервис, с помощью которого есть возможность кэшировать различные пары ключ-значение в базе данных Redis.

9 Используемые технические средства

Программная DATASKAI поддерживает x86_64—совместимые аппаратные DATASKAI. Данные DATASKAI выбраны как наиболее массовые и распространенные среди производителей аппаратных вычислительных средств.

В зависимости от объемов обрабатываемых данных и требуемых объемов вычислений DATASKAI поддерживает развертывание на различном количестве аппаратных средств, как на одиночном сервере, так и на наборе серверов. При развертывании на наборе серверов в качестве сетевых интерфейсов используется технология Ethernet.

Минимальные требования при развертывании на одиночном сервере:

— ЦП x86_64 архитектура, по производительности не ниже Intel Xeon Silver 4116T Processor, кол—во вычислительных ядер не менее 12;

— ОЗУ не менее 192 Гб;

— ПЗУ не менее 500 Гб;

— сетевой адаптер 1 Гб.

Рекомендуемые требования при развертывании на одиночном сервере:

— ЦП x86_64 архитектура, Intel Xeon Platinum 8176 Processor, кол—во вычислительных ядер 28; — ОЗУ 256 Гб;

— ПЗУ 3 Тб;

Сетевой адаптер 10Gb.

10 Вызов и загрузка

10.1 Способ вызова программы с соответствующего носителя данных

Развертывание и запуск компонентов DATASKAI осуществляются в ручном режиме или с помощью автоматизированной процедуры на основе технологий Docker, Docker compose, Kubernetes. Подобные инструкции и порядок развертывания компонентов DATASKAI содержатся в руководстве администратора. В процессе работы запущенные компоненты функционируют в фоновом режиме. Поддерживаются запуск и остановка отдельных компонентов для их обновления или переконфигурирования без необходимости остановки всего комплекса.

Возможна поставка продукта в виде, допускающем развертывание в изолированной среде.

10.2 Входные точки программного комплекса

Поступающие непрерывным потоком неструктурированные данные через модули импорта данных попадают в шлюз данных, через который перенаправляются в шину данных. Данные из шины преобразуются модулями ETL во внутреннее структурированное представление и загружаются в подсистему хранения данных.

Подсистема хранения данных через каталог данных предоставляет метаинформацию о хранимых данных, через API позволяет запрашивать необходимые данные компонентам ПК и внешним клиентам. Внутреннее хранение данных реализовано согласно их типам в нескольких типах хранилищ, таких как реляционные СУБД, нереляционные СУБД, хранилища временных рядов, файловые хранилища и т.п.

11 Входные данные

11.1 Характер, организация и предварительная подготовка входных данных

Подсистема сбора и импорта данных поддерживает два режима сбора входных данных:

— активный опрос источника данных модулем импорта;

— пассивный приём через шлюз данных от внешних клиентов.

11.2 Формат, описание и способ кодирования входных данных

Входными данными для продукта являются данные из информационных систем эксплуатанта DATASKAI, справочников, внешних источников, к которым подключены модули импорта данных. DATASKAI не накладывает жестких требований на форматы входных данных, кроме требований машиночитаемости формата.

Интеграция с различными типами источников и форматов данных осуществляется на уровне подсистемы сбора и импорта данных DATASKAI. Такой подход позволяет обеспечить максимальную гибкость при подключении произвольных источников данных. Кроме того, для распространенных типов источников, например, реляционных СУБД, в рамках программного продукта предусмотрены готовые инструменты для реализации модулей импорта.

12 Выходные данные

12.1 Характер и организация выходных данных

Доступ к выходным данным предоставляется через открытые интерфейсы программной DATASKAI, работающие поверх подсистемы хранения данных. Одним из поддерживаемых механизмов доступа к выходным данным является протокол HTTP.

В качестве основных форматов выходных данных используются CSV и JSON форматы. Данный подход является общепринятым для реализации внешних интерфейсов доступа к данным.

12.2 Формат, описание и способ кодирования выходных данных

Выходными данными программной DATASKAI являются:

— первичные данные собранные из входных источников;

— признаки объектов предметной области, рассчитанные на первичных данных;

— результаты работы алгоритмов принятия решений/моделей на признаках объектов.

DATASKAI должна выполнять следующие основные функции:

– сбор данных из источников данных и их преобразование во внутренние представления для эффективного хранения, обработки и анализа;

– хранение внутренних представлений данных и реализация унифицированного доступа к данным через внешние интерфейсы;

– обработку данных с возможностью встраивания экспертных моделей предметной области;

– анализ с возможностью формализации постановок и контроля решения различных классов задач на данных.

13 Характеристики пользователя

Пользователями DATASKAI должны являться:

– специалист по обработке данных, решающий задачи анализа данных с использованием DATASKAI;

– разработчик внешних приложений, реализующий приложения для конечных потребителей результатов решения задач анализа данных;

– системный администратор, осуществляющий развертывание и сопровождение DATASKAI;

– инженер по данным, решающий задачи по организации сбора и хранения данных с использованием DATASKAI.

14 Ограничения

С учетом специфики применения DATASKAI ее реализация имеет следующие характерные особенности:

– ориентация на работу с данными, зависящими от времени, а также табличными данными, связанными реляционными отношениями;

– не предполагается работа с данными, хранимыми в машиночитаемом графическом виде, однако с заложенной возможностью расширения на другие типы данных с использованием открытого API;

– не предполагается работа с данными в режиме реального времени, вместо этого DATASKAI нацелена на работу с данными в потоковом режиме;

– поток входящих данных для сбора и хранения не более 1ТБ / сутки.