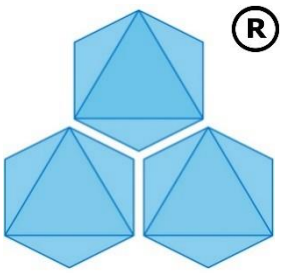

КРАММЕРТИ

Описание технической архитектуры программного обеспечения

Программа для ЭВМ
«Краммерти. Модуль прогнозирования временных рядов»

1. Описание

Программа для ЭВМ «Краммерти. Модуль прогнозирования временных рядов» предназначена для автоматизированного краткосрочного прогнозирования временных рядов метеорологических параметров (и не только). Разработка данного модуля произведена в рамках развития проекта по прогнозированию температуры воздуха и поставляется как в составе «Краммерти. Подсистема метеомониторинга» так и как отдельный программный продукт в зависимости от Задач.

Программа разработана для нужд оперативного мониторинга и планирования, предоставляя как основной прогноз, так и расчетную температуру "по ощущениям" (с учетом ветро-холодового индекса). Программа интегрирует данные из внутреннего и/или внешнего API исторических наблюдений необходимых параметров (метеостанции, и иное имеющееся на балансе Заказчика измерительное оборудование и т.д. временные ряды) а также учитываются внешние прогностические данные сторонних сервисов (если это необходимо).

Применяемая модель: LSTM (Long Short Term Memory) — рекуррентная нейронная сеть глубокого обучения, которая способна улавливать закономерности в данных временных рядов и использоваться для прогнозирования будущего тренда данных.

Система автоматизирует полный цикл прогнозирования включая:

- **Сбор и обновление данных:**

o Автоматическая загрузка исторических данных по заданным параметрам (напр., температура воздуха) с указанных метеостанций через внутренний API (с обработкой пагинации, кэшированием, повторными попытками и обработкой ошибок).

o Проверка свежести локально сохраненных исторических данных и дозагрузка только новых записей для минимизации трафика и времени.

o Автоматическая загрузка актуальных прогнозов (температура, скорость ветра и/или иных доступных данных временных рядов) от внешних прогностических сервисов.

o Проверка свежести локальных копий данных от прогностических сервисов для сокращения запросов к внешнему API.

- **Хранение данных: Сохранение загруженных исторических данных и прогнозов в локальные CSV-файлы для последующего использования Системой.**
- **Предобработка данных:**

o Очистка данных: Удаление выбросов методом IQR.

o Ресемплинг: Приведение данных к единой временной сетке (по умолчанию - часовой).

o Инжиниринг признаков: Создание лаговых, календарных (час, день года, месяц, день недели и т.д.) и циклических (sin/cos) признаков для модели LSTM.

o Масштабирование: Нормализация данных с помощью RobustScaler для подготовки к подаче в нейронную сеть.

- **Подбор гиперпараметров модели нейросети LSTM методом HPO (Hyper Parameter Optimization):**
  - Использование фреймворка Optuna для автоматического поиска оптимальной комбинации гиперпараметров LSTM-модели (количество нейронов, скорость обучения, функции активации, параметры регуляризации, длина входной последовательности и т.д.) на основе заданных диапазонов в hyperparameters.json.
  - Применение (на этапе HPO) функции потерь, которая учитывает отклонение от фактических данных.
- **Обучение модели:**
  - Обучение финальной LSTM-модели на предобработанных исторических данных с использованием лучших гиперпараметров, найденных Optuna.
  - Использование стандартной функции потерь Huber для обучения финальной модели (не зависит от сторонних прогностических данных на этом этапе).
  - Применение механизма EarlyStopping для предотвращения переобучения.
- **Сохранение и версионирование модели:**
  - Сохранение обученной модели Keras (.keras), соответствующего объекта Scaler (.pkl) и метаданных (включая гиперпараметры и метрики) в JSON-файл.
  - Включение метрики SMAPE и даты в имя файла для отслеживания производительности и выбора лучшей модели.
- **Генерация прогноза:**
  - Загрузка лучшей (по SMAPE) сохраненной модели и скейлера для целевого параметра.
  - Генерация прогноза температуры на заданный горизонт с помощью LSTM-модели.
  - Загрузка актуального прогноза прогностических сервисов (температура и ветер).
  - Коррекция прогноза: контроль корректности прогноза LSTM путем его сравнения с прогнозами прогностических сервисов, регулирование расхождения при превышении заданного порога (CORRECTION_THRESHOLD), с учетом коэффициента (CORRECTION_FACTOR) и максимального ограничения (MAX_CORRECTION).
  - Расчет Wind Chill: Вычисление температуры "по ощущениям" на основе скорректированного прогноза температуры и прогноза скорости ветра прогностических сервисов.
- **Вывод результатов:**
  - Сохранение итогового прогноза (включая температуру и "ощущаемую" температуру) в структурированный JSON-файл с временными метками.
  - Генерация и сохранение графика, визуализирующего исторические данные, прогноз LSTM (скорректированный), прогноз "по ощущениям" и прогноз(ы) прогностического сервиса (ов).

- **Автоматизация:**
  - Использование библиотеки schedule для запуска задач по расписанию (обновление данных, переобучение модели, генерация прогноза).
  - Возможность интеграции с systemd (на Linux) для обеспечения постоянной работы и автоматического перезапуска сервисов (обновления данных/планировщика и API).
- **API для доступа к прогнозу:**
  - Предоставление простого веб-API (на базе Flask), которое по запросу отдает содержимое самого свежего сгенерированного JSON-файла с прогнозом.

1.1. Список используемых библиотек и компонентов разработке и функционировании продукта, а также тип лицензии к ним.

Категория 1: Наука о данных и Машинное обучение

Это ядро проекта, отвечающее за обработку данных и построение модели.

БИБЛИОТЕКА	ПРЕДНАЗНАЧЕНИЕ	ИСПОЛЬЗОВАНИЕ В ПРОЕКТЕ	ЛИЦЕНЗИЯ
TENSORFLOW (`TENSORFLOW`)	Открытая платформа для машинного обучения от Google.	**Это ядро всей системы.** Используется для построения, компиляции, обучения (`model.fit`), предсказания (`model.predict`) и сохранения/загрузки (`.keras`) LSTM-модели. (`lstm_model.py`, `main.py`)	Apache 2.0
PANDAS (`PANDAS`)	Библиотека для обработки и анализа данных, предоставляющая структуру DataFrame.	**Ключевой инструмент для данных.** Используется для чтения CSV (`pd.read_csv`), манипуляции с данными, работы с временными рядами (ресемплинг, индексация), объединения данных. (`data_loader.py`, `data_preprocessing.py`, `main.py`)	BSD 3-Clause
NUMPY (`NUMPY`)	Фундаментальный пакет для научных вычислений.	**Основа для всех числовых операций.** Используется для создания массивов, математических операций (sin, cos для признаков), изменения формы данных для подачи в LSTM, замены NaN. (`data_preprocessing.py`, `lstm_model.py`, `metrics.py`, `main.py`)	BSD 3-Clause
SCIKIT-LEARN (`SCIKIT-LEARN`)	Инструменты для анализа данных и машинного обучения.	Используется для: • **Масштабирования данных** (`RobustScaler`). • **Разделения выборки** на обучающую и валидационную (`train_test_split`). • **Расчета метрик** (`mean_absolute_error`, и т.д.).	BSD 3-Clause

		(`data_preprocessing.py`, `metrics.py`, `main.py`)	
OPTUNA (`OPTUNA`)	Фреймворк для автоматической оптимизации гиперпараметров.	**Автоматически подбирает лучшие параметры для LSTM-модели** (количество нейронов, скорость обучения и др.), что значительно повышает ее качество. (`main.py`)	MIT License
SCIPY (`SCIPY`)	Библиотека для научных и инженерных вычислений.	Используется для **Гауссова сглаживания** прогнозов (`gaussian_filter`) в модуле предобработки, хотя этот метод в итоге не вызывается в основном цикле. (`data_preprocessing.py`)	BSD 3-Clause
MATPLOTLIB (`MATPLOTLIB`)	Библиотека для создания статических, анимированных и интерактивных визуализаций.	Используется для **создания графиков**, которые показывают сравнение исторических данных, прогноза LSTM и прогноза Open-Meteo. Графики сохраняются в папку `graphs/`. (`main.py`)	PSF-based (BSD-like)

Категория 2: Веб, API и Сетевые взаимодействия

Эта группа отвечает за получение данных из внешних источников и предоставление результата.

БИБЛИОТЕКА	ПРЕДНАЗНАЧЕНИЕ	ИСПОЛЬЗОВАНИЕ В ПРОЕКТЕ	ЛИЦЕНЗИЯ
FLASK (`FLASK`)	Микрофреймворк для создания веб-приложений на Python.	**Создает API-сервер** (`api_server.py`) с эндпоинтом, который отдает самый свежий JSON-файл с прогнозом.	BSD 3-Clause
REQUESTS (`REQUESTS`)	Элегантная и простая HTTP-библиотека для людей.	Используется для **синхронных HTTP GET-запросов** к API Open-Meteo и, как запасной вариант, в `ApiClient`. (`api_client.py`, `open_meteo_client.py`)	Apache 2.0
AIOHTTP (`AIOHTTP`)	Библиотека для асинхронных HTTP-	Используется для **быстрых асинхронных запросов** к API `api-meteor.admsurgut.ru` для загрузки	Apache 2.0

	клиентов/серверов.	больших объемов исторических данных. (`api_client.py`)	
TENACITY (`TENACITY`)	Библиотека для реализации логики повторных попыток (retry).	**Повышает надежность системы.** Автоматически повторяет сетевые запросы к обоим API в случае временных сбоев или ошибок. (`api_client.py`, `open_meteo_client.py`)	Apache 2.0
REQUESTS-CACHE	Прозрачное кеширование для библиотеки `requests`.	**Ускоряет повторные запросы** к API, сохраняя ответы в локальный файл `api_cache.sqlite`. Включается/выключается через `config.ini`. (`api_client.py`)	BSD 2-Clause

Категория 3: Вспомогательные и системные библиотеки

Эти библиотеки выполняют служебные функции: работа с файлами, временем, планирование задач и т.д.

БИБЛИОТЕКА	ПРЕДНАЗНАЧЕНИЕ	ИСПОЛЬЗОВАНИЕ В ПРОЕКТЕ	ЛИЦЕНЗИЯ
SCHEDULE (`SCHEDULE`)	Простая библиотека для планирования периодических задач.	**Оркестратор всей автоматизации.** Запускает по расписанию (`config.ini`) задачи обновления данных, обучения модели и создания прогнозов. (`main.py`)	MIT License
CONFIGPARSER	Встроенная библиотека для работы с конфигурационными файлами.	**Читает все настройки** из файла `config.ini`, делая проект гибким и легко настраиваемым без изменения кода. (Все модули)	Python (PSF)
TQDM (`TQDM`)	Библиотека для создания умных и расширяемых прогресс-баров.	**Визуализирует прогресс** загрузки данных при асинхронных запросах с пагинацией, показывая, сколько страниц уже загружено. (`api_client.py`)	MIT/MPL 2.0
PYTHON-DATEUTIL	Мощные расширения для	Используется для **надежного парсинга дат** в различных форматах	Apache/BSD

	стандартного модуля `datetime`.	из ответов API (`isoparse`). (`main.py`)	
OS, GLOB, RE, JSON, CSV, PICKLE, TIME, DATETIME, THREADING, ASYNCIO, MATH, LOGGING, PATHLIB	Встроенные библиотеки Python.	• `os, glob, pathlib, re`: Работа с файловой системой, путями, поиск файлов по шаблону, создание безопасных имен файлов. • `json, csv, pickle`: (Де)сериализация данных: метаданные, параметры, метрики, скейлеры. • `time, datetime`: Все операции со временем и датами. • `threading, asyncio`: Управление параллелизмом и асинхронностью. • `math`: Для математических расчетов (Wind Chill). • `logging`: Ведение логов работы приложения.	Python (PSF)

2. Архитектура

Модуль представляет собой модульное Python-приложение, состоящее из следующих ключевых компонентов:

- **Модули доступа к данным:**
  - **api_client.py**: взаимодействует с внутренним API исторических данных.
  - **api_server.py**: взаимодействует с внешними потребителями прогностических данных предоставляя API интеграцию.
  - **open_meteo_client.py**: Взаимодействует с источниками внешних API прогностических сервисов.
  - **data_loader.py**: загружает данные из локальных CSV-файлов.
- **Модуль предобработки:**
  - **data_preprocessing.py**: реализует шаги очистки, ресемплинга, создания признаков и масштабирования данных.
- **Модуль моделирования:**
  - **lstm_model.py**: содержит определение архитектуры LSTM-модели, логику ее компиляции (с возможностью использования кастомных или стандартных потерь/метрик), методы для обучения, предсказания, сохранения и загрузки.
  - **metrics.py**: Функции для расчета метрик качества (MAE, RMSE, SMAPE и т.д.).

- **Модуль оркестрации и логики:**
    - **main.py**: является точкой входа, инициализирует компоненты, читает конфигурацию, содержит логику HPO (Optuna), запускает процессы обучения и прогнозирования, управляет планировщиком задач (schedule) и содержит вспомогательные функции (например, calculate_wind_chill, get_best_model).

- **Модуль API-сервера:**
    - **api_server.py**: реализует веб-сервер Flask, который предоставляет эндпойнт для получения последнего сгенерированного прогноза из JSON-файла.

- **Конфигурационные файлы:**
    - config.ini: хранит все основные настройки системы (пути, URL, токены, параметры API, расписание, параметры коррекции и т.д.).
    - hyperparameters.json: Определяет диапазоны и типы гиперпараметров для поиска лучших настроек модели LSTM нейросети фреймворком Optuna.
- **Хранилище данных/артефактов:**
    - **Папки (weather_data, models, logs, graphs, lstm_forecasts):** используются для хранения CSV-данных, обученных моделей, логов выполнения обучения, графиков и JSON-прогнозов соответственно, предметно:

      weather_data – хранение исторических данных параметров например Температуры воздуха.csv за весь возможный период доступной истории для каждого источника дифференцировано.

      models – обученные модели .keras, гиперпараметры в .json обученных моделей, скейлеры моделей .pkl.

      logs – каталог хранения базы данных оптимизации и показателей параметров при HPO (hyper parameter optimization) при обучении моделей.

      graphs – каталог хранения графиков обучения и прогнозов.

      lstm_forecasts – каталог хранения JSON файлов прогнозов которые в последующем передаются по API во внешние системы.
    - ***База данных Optuna** (.db файл в папке models): хранит историю и результаты HPO (hyper parameter optimization — это процесс настройки параметров модели для достижения оптимальной производительности. Он необходим для оценки множества вариантов дизайна и справедливого сравнения моделей.

5. Описание файлов Python (.py):

- **main.py:** Главный скрипт, "дирижер" всего оркестра. Отвечает за:
    - Чтение конфигураций (config.ini, hyperparameters.json).
    - Инициализацию основных объектов (API клиентов, загрузчиков, препроцессоров).
    - Оркестрацию первоначальной загрузки данных.
    - Запуск процесса HPO и обучения финальной модели (train_lstm_model), вызывая функции из других модулей.

    - Запуск процесса генерации прогноза (run_prediction_tasks), вызывая get_best_model и make_forecast.

    - Настройку и запуск планировщика schedule с задачами (job_*).
    - Содержит вспомогательные функции, такие как calculate_wind_chill, get_best_model, save_metrics, create_safe_filename.
- **api_client.py:** реализует класс ApiClient для взаимодействия с внутренним API. Обрабатывает авторизацию (токен), таймауты, повторные попытки (tenacity), кэширование (requests-cache), пагинацию (в async_fetch_historical_data).
- **open_meteo_client.py:** реализует класс OpenMeteoClient для взаимодействия с прогностическими сервисами API. Запрашивает прогноз температуры и ветра, обрабатывает таймауты и повторные попытки, извлекает данные (extract_forecast_data).
- **data_loader.py:** реализует класс DataLoader, отвечающий за безопасную загрузку данных из локальных CSV-файлов с использованием Pandas, обработку ошибок чтения и парсинга дат.
- **data_preprocessing.py:** реализует класс DataPreprocessor. Содержит методы для:
    - Удаления выбросов (remove_outliers).
    - Ресемплинга данных (resample_data).
    - Создания признаков (create_features), включая лаги и календарные/циклические признаки.
    - Масштабирования данных с помощью RobustScaler (fit_transform, transform, inverse_transform).
    - (Опционально) Сглаживания данных (smooth_data).
- **lstm_model.py:** реализует класс LSTMModel. Отвечает за:
    - Определение архитектуры нейронной сети LSTM с использованием TensorFlow/Keras.
    - Компиляцию модели с оптимизатором (Adam) и функцией потерь (стандартный Huber или кастомный) и метриками (стандартная MAE или MaskedMAE).
    - Метод train для обучения модели на подготовленных данных, включая использование коллбэков (EarlyStopping, TensorBoard).
    - Метод predict для генерации прогнозов на новых данных.
    - Методы save и load для сохранения/загрузки весов модели Keras.
    - Статический метод prepare_data для преобразования временного ряда в формат (X, y) для LSTM.
- **metrics.py:** содержит функции для расчета метрик качества прогноза (MAE, RMSE, MAPE, R2, SMAPE) метрики сохраняются в metrics.csv файл локально при каждом обучении моделей.

api_server.py: реализует веб-сервер на Flask, который находит последний (самый свежий - сегодняшний) JSON-файл прогноза и отдает его содержимое по GET запросу.

Установка и эксплуатация

Установка и эксплуатация предполагает использование linux-подобной операционной системы и наличие следующих необходимых компонентов.

Требования к серверному оборудованию (установка - развертывание программы на серверах Заказчика)*

№ п/п	Наименование виртуальной машины (прим.)	Количество ядер	RAM, Гб	SSD, Гб	ОС
1.	ai_its	12+	16+	200	Ubuntu 20 v+

*Подробное описание минимальных и рекомендуемых конфигураций содержится в документации к настоящему программному обеспечению «Инструкция по установке».