ВИЛИОН

ПРОГРАММНЫЙ ПРОДУКТ

ОПИСАНИЕ ПРОГРАММА ДЛЯ ЭВМ «ПРЕДИКТИВНОЕ ОПОВЕЩЕНИЕ О ЧРЕЗВЫЧАЙНЫХ СИТУАЦИЯХ И ИХ ПОСЛЕДСТВИЯХ»

Краснодар
2022

Оглавление

1. Назначение

Настоящий документ содержит описание программного обеспечения «Предиктивное оповещение о чрезвычайных ситуациях и их последствиях», а также описывает его архитектуру.

2. Введение

2.1. Назначение программы для ЭВМ

Программное обеспечение для предиктивного оповещения о чрезвычайных ситуациях и их последствиях на базе математических моделей и машинного обучения. Предиктивный анализ производится на основе данных, поступающих из внешних источников (датчики, сводки и т.п.).

2.2. Реферат

ПО работает в качестве модуля маршрутизации потока данных (событий) для обеспечения работоспособности предиктивной аналитики. Прогнозирование будущих событий осуществляется на основе поступающих (текущих) событий, а также имеющихся исторических событий. Маршрутизация событий осуществляется между источниками событий, модулями прогнозирования и потребителями; маршрутизация осуществляется согласно настройкам, по заранее заданным шаблонам. В ПО непосредственно входят два модуля прогнозирования: модуль «Моделирование распространения инфекционных заболеваний» и модуль «Прогнозирование вероятности наводнений».

2.3. Языки программирования

Используемый язык программирования – Python 3.

2.4. Объем дистрибутива ПО

Объем дистрибутива ПО «Предиктивное оповещение о чрезвычайных ситуациях и их последствиях» - 400МБ.

3. Описание ПО

ПО «Предиктивное оповещение о чрезвычайных ситуациях и их последствиях» предназначено для предиктивного оповещения о чрезвычайных ситуациях и их последствиях на базе математических моделей и машинного обучения. Предиктивный анализ производится на основе данных, поступающих из внешних источников (датчики, сводки и т.п.)

ПО работает в составе отдельных модулей, предназначенных для маршрутизации потока данных (событий), запуска отдельных моделей предиктивной аналитики, а также для контроля и маркировки прогнозных значений.

Прогнозирование будущих событий осуществляется на основе поступающих (текущих) событий, а также имеющихся исторических событий.

Маршрутизация событий осуществляется между источниками событий, модулями прогнозирования и потребителями; маршрутизация осуществляется согласно настройкам.

4. Архитектура ПО

Архитектура ПО представляет собой набор независимо исполняемых модулей, интегрированных друг с другом через очереди брокера событий. Реализованы следующие модули:

- API подключения датчиков — модуль, реализующий интеграцию с внешними датчиками посредством REST HTTP API. Принимает входящие показания с датчиков и размещает данные во входящую очередь событий 'sensors'.
- Модуль ручной загрузки — предназначен для загрузки данных в ручном режиме — применяется для загрузки архивных данных, а также в случае отсутствия возможности прямой интеграции с датчиками. Принимает входящие показания из файлов и размещает данные во входящую очередь событий 'sensors'.
- Модуль сортировки — согласно правилам сортировки, определяемым через конфигурационный файл, производит сортировку поступающих во входную очередь данных в соответствующие очереди для алгоритма прогноза. Отбор осуществляется в соответствии с идентификатором датчика и имеющимся алгоритмом прогнозирования.
- Модули прогноза — модули, непосредственно реализующие прогнозные модели. Поддерживаются как математические модели, так и модели машинного обучения. Каждый модуль подключен к выделенной очереди входящих событий и формирует прогноз так же в выделенную очередь прогнозных значений.
- Модуль классификации уровня опасности — формирует уровень опасности (по умолчанию уровня три: «зеленый», «желтый», «красный») в соответствии с прогнозными значениями и фактически

имеющимися данными по гидрологическому посту. Критерий определяется настройками, которые привязаны к идентификатору датчика.

- Модуль определения зоны подтопления — формирует перечень зон подтопления в районе гидрологического поста в соответствии с корреляционными таблицами географических зон и относительных показаний датчиков.
- API получения результатов - модуль, реализующий интеграцию с внешними получателями прогноза посредством REST HTTP API. Возвращает промаркированный уровнем опасности прогноз в соответствии с идентификатором датчика исходного события.

Архитектура модулей ПО представлена на рисунке 1.

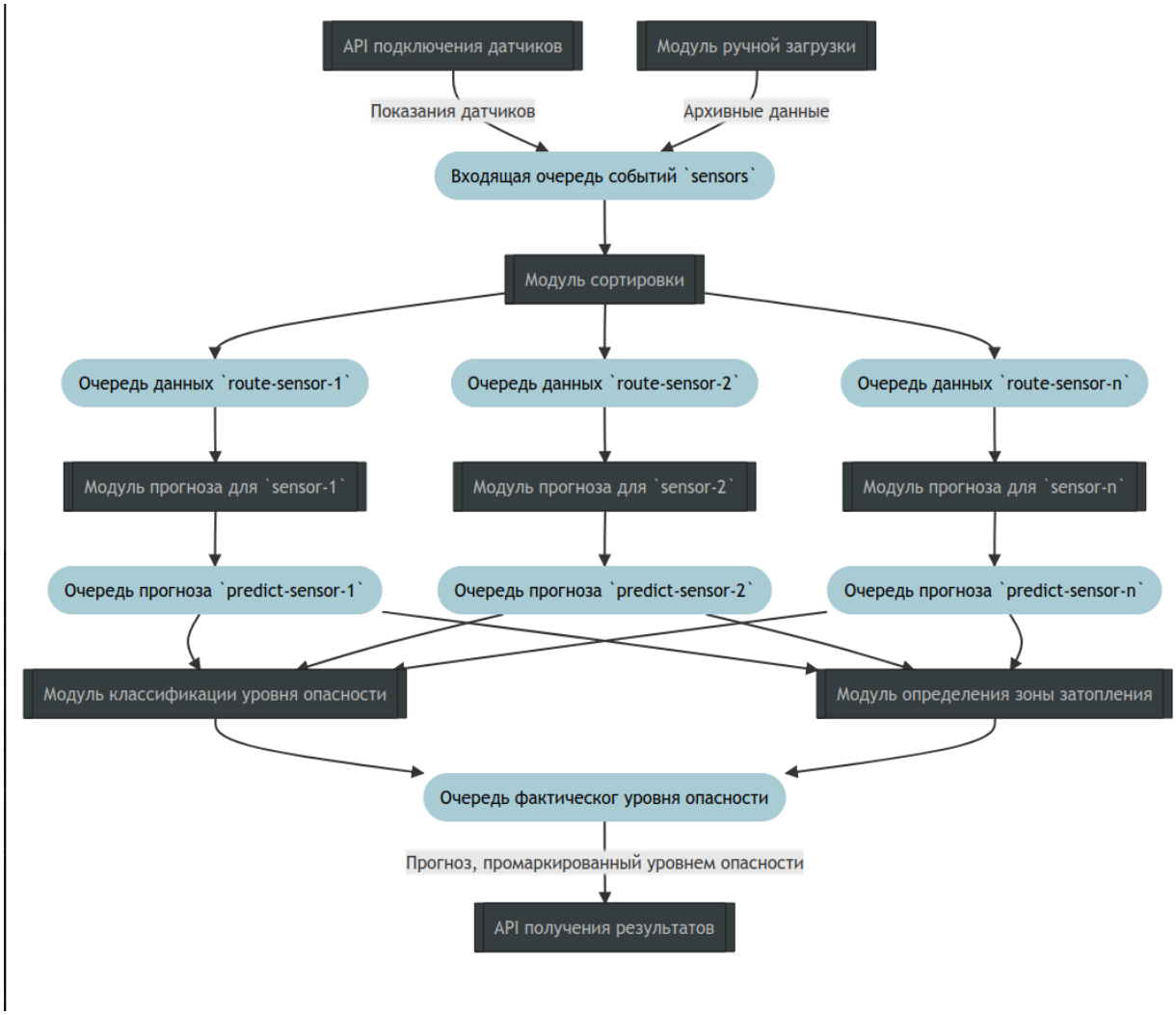

Рисунок 1 – Архитектура модулей ПО

5. Архитектура данных

Для организации внутреннего обмена между модулями и персистентного хранения информации используется низкоуровневый формат сериализации данных Apache Avro.

В соответствии с форматом Avro, произведено проектирование следующих универсальных моделей данных:

- Datum — универсальный заголовок набора данных
- Grain — фактически собранные данные (элементарная единица)
- Dish — результат прогнозирования (элементарная единица)

- Species — описание набора данных
- Family — тип набора данных (перечисление)
- Molecule — состав элементарной единицы в наборе данных (перечисление)
- Quant — квант элементарной единицы в наборе данных (перечисление)

Ниже подробно описана каждая модель в формате AVRO.

Datum — *универсальный заголовок набора данных*

```
{
   "name": "Datum",
   "doc": "Header of dataset",

   "type": "record",
   "fields": [
      {
         "name": "epoch",
         "doc": "Datetime",

         "type": {"type": "long", "logicalType": "timestamp-millis"}
      },
      {
         "name": "name",
         "doc": "Uniq datasource identification string (Species.name)",

         "type": "string"
```

```
        }
      ]
    }
```

***Grain** — фактически собранные данные (элементарная единица)*

```
    {
      "name": "Grain",
      "doc": "Single element of dataset",

      "type": "record",
      "fields": [
        {
          "name": "epoch",
          "doc": "Datetime (optional)",

          "type": ["null", {"type": "long", "logicalType": "timestamp-millis"}]
        },
        {
          "name": "x",
          "doc": "Source values array",

          "type": {"type": "array", "items": ["null", "double"]
                }
        },
        {
          "name": "y",
```

```
                "doc": "Actual values array",

                "type": {"type": "array", "items": ["null", "double"]}
            }
        ]
    }
```

Dish — *результат прогнозирования (элементарная единица)*

```
    {
        "name": "Dish",
        "doc": "Predicted dataset",

        "type": "record",
        "fields": [
            {
                "name": "epoch",
                "type": ["null", {"type": "long", "logicalType": "timestamp-millis"}]
            },
            {
                "name": "r",
                "doc": "Prediction result",

                "type": {"type": "array", "items": "double"}
            },
            {
                "name": "p",
```

```
            "doc": "Probability of result (optional)",


            "type": {"type": "array", "items": ["null", "double"]}
        }
    ]
}
```

Species — *описание набора данных*

```
{
    "namespace": "ru.vilion-k.model",
    "type": "record",
    "name": "Species",
    "doc": "Datasource definition",
    "fields": [
        {
            "name": "name",
            "type": {
                "type": "string",
            },
            "doc": "Uniq datasource identification string"
        },
        {
            "name": "desc",
            "type": {
                "type": "string",
            },
            "doc": "Extended description of datasource"
```

```
},
{
  "name": "family",
  "type": {
    "type": "enum",
    "name": "Family",
    "doc": "Dataset type",
    "symbols": [
      "generic",
      "water_level",
      "disease_spread"
    ],
    "default": "generic"
  },
  "doc": "Datasource type",
  "default": "generic"
},
{
  "name": "molecule",
  "type": {
    "type": "enum",
    "name": "Molecule",
    "doc": "Dataset scope",
    "symbols": [
      "atom",
      "raw",
      "timeseries"
    ],
```

```
      "default": "atom"
    },
    "doc": "Typical dataset scope",
    "default": "atom"
  },
  {
    "name": "quant",
    "type": {
      "type": "enum",
      "name": "Quant",
      "doc": "Dataset quant",
      "symbols": [
        "none",
        "year",
        "quarter",
        "month",
        "week",
        "day",
        "hour",
        "halfhour",
        "quarterhour",
        "minute",
        "second"
      ],
      "default": "none"
    },
    "doc": "Dataset quant",
    "default": "none"
```

```
},
{
  "name": "lengths",
  "type": {
    "type": "record",
    "name": "Lengths",
    "fields": [
      {
        "name": "x",
        "type": "long",
        "default": 0
      },
      {
        "name": "y",
        "type": "long",
        "default": 0
      },
      {
        "name": "r",
        "type": "long",
        "default": 0
      },
      {
        "name": "p",
        "type": "long",
        "default": 0
      }
    ]
```

```
            },
            "doc": "Default length of arrays"
        }
    ]
}
```

Family — *тип набора данных (перечисление)*

```
{
    "namespace": "ru.vilion-k.model",
    "name": "Family",
    "doc": "Dataset type",

    "type": "enum",
    "symbols": [
        "generic", "water_level", "disease_spread"
    ],
    "default": "generic"
}
```

Molecule — *состав элементарной единицы в наборе данных (перечисление)*

```
{
    "namespace": "ru.vilion-k.model",
    "name": "Molecule",
    "doc": "Dataset scope",

    "type": "enum",
```

```
    "symbols": [
      "atom", "raw", "timeseries"
    ],
    "default": "atom"
  }
```

Quant — *квант элементарной единицы в наборе данных (перечисление)*

```
  {
    "namespace": "ru.vilion-k.model",
    "name": "Quant",
    "doc": "Dataset quant",

    "type": "enum",
    "symbols": [
      "none", "year", "quarter", "month", "week", "day",
"hour", "halfhour", "quarterhour", "minute",
      "second"
    ],
    "default": "none"
  }
```

6. Техническая архитектура

Общие модули реализованы на языке программирования Python 3.9 с использованием библиотек faust-streaming, fastavro, aiohttp, python-avro, confluent-kafka.

Интеграция осуществляется через распределенный брокер событий kafka 3.

Работоспособность протестирована в ОС Linux Ubuntu 21.10.

7. Перечень сокращений

ОС — Операционная система

ПО — Программное обеспечение

API — программный интерфейс приложения представляет собой набор правил, определяющих способ взаимодействия между приложениями или устройствами.