## Задача 3. Прогнозирование потребления тепла

В уездном городке У есть проблема: в конце каждой осени приходит зима. А с зимой приходят холода. А когда на улице холодно, жители просят, чтобы дома было тепло. При этом каждый дом потребляет различное количество теплоты, и хотелось бы спрогнозировать, какое же будет потребление по дому в предстоящую зиму.

В большинстве домов городка установлены счётчики теплоты, которые отдают в автоматическом режиме данные каждый день в течение всего отопительного сезона.

Необходимо спрогнозировать суммарное потребление теплоты каждым домом на **следующий отопительный сезон** (с 01.10.2022 по 30.04.2023), чтобы городской поставщик отопления был уверен, что ресурсов у него хватит.

Данные для построения модели:

1. Архив температуры наружного воздуха (файл «*температура.csv*») со следующими параметрами:
   * *date\_start* - дата и время начала периода измерений;
   * *date\_end* - дата и время окончания периода измерений;
   * *temp* - средняя температура за период;
   * *temp\_max* - наибольшая температура за период;
   * *temp\_min* - наименьшая температура за период;
   * *humidity* - влажность.
2. Характеристики домов (файл «*характеристики\_домов.csv*») со следующими параметрами:
   * *address\_uuid* - уникальный идентификатор адреса;
   * *build\_year* - год постройки;
   * *floor\_number* - количество этажей;
   * *residential\_area* - общая жилая площадь;
   * *wall\_type* - тип наружных стен;
   * *roof\_area\_total* - общая площадь крыши
   * *roof\_area\_metal* - площадь металлической кровли ;
   * *roof\_area\_web* - площадь кровли из рулонных материалов;
   * *roof\_area\_piece\_goods* - площадь кровли из штучных материалов.
3. Суточных архив показаний тепловычислителей (файл «*объём\_теплоты.csv*») со следующими параметрами;
   * *address\_uuid* - уникальный идентификатор адреса;
   * *date* - дата, за которую снимается показание;
   * *value* - значение потреблённой домом теплоты за указанную дату;
   * *is\_unreliable* - флаг достоверности показаний.

Исходные данные:

1. Файл с «прогнозом» температуры на следующий отопительный сезон ***/mnt/source/temperature.csv*** с параметрами, аналогичными файлу «*температура.csv*» из данных для построения модели.

Выходные данные:

1. Файл с суммой потреблённого с 01.10.2022 по 30.04.2023 тепла по каждому дому ***/mnt/result/volume.csv*** со следующими параметрами:
   * *address\_uuid* - уникальный идентификатор адреса;
   * *volume* - сумма спрогнозированных показаний по дому по датам из исходных данных на основании указанного прогноза погоды.

Примечания:

* Формат всех файлов CSV: с заголовком, разделители - «;».
* Считается, что показания «прогноза» температуры относятся к отопительному сезону, следующему за данными для построения модели.
* Дополнительные данные о домах даны, чтобы можно было кластеризовать временные ряды показаний потребления тепла по характеристикам домов.

На проверку отправляется архив, в котором присутствует:

* Файл ***README.pdf*** с кратким описанием применённого решения.
* Исходные тексты программы, формирующей модель, в папке ***/src-train***.
* Исходные тексты программы, преобразующей исходные данные в выходные на основании заранее сформированной (обученной) модели в папке ***/src-predict***.
* Собственно модель в папке ***/model***.
* Файл ***Dockerfile***, в котором создаётся контейнер и запускается программа для преобразования исходных данных в выходные (из */src-predict*).

Требование самостоятельно описывать окружение через Dockerfile необходимо для унификации проверки. Пример файла «*docker-compose.yml*» для запуска задания на проверку:

|  |
| --- |
| version: "3.1"  services:  app:  build: .  volumes:  - .:/app  - ${PATH\_SOURCE}:/mnt/source  - ${PATH\_RESULT}:/mnt/result |

, где ${PATH\_SOURCE} - путь до каталога с исходными данными, ${PATH\_RESULT} - путь до каталога с выходными данными для проверки.

То есть запуск программы будет осуществляться со следующими характеристиками:

* ОС - Linux, настроенная через Dockerfile для запуска программы;
* Данные их архива на проверку примонтированы в */app*;
* Соответственно, модель находится по пути */app/model*;
* Исходные данные примонтированы в */mnt/source*;
* Каталог для сбора выходных данных примонтирован в */mnt/result*.

Запуск контейнера будет осуществляться через команду «docker compose up --build» с заданными переменными окружения PATH\_SOURCE и PATH\_RESULT.

**Если у Вас возникают трудности с написанием Dockerfile, опишите более подробно, как запустить программу predict для формирования выходного файла.**