## Задача 2. Распознавание показаний приборов учёта

В уездном городке М возникла проблема: там очень хорошо работает ЖКХ. До того хорошо, что жители просят распознавать показания своих счётчиков по фотографиям. Например, включил на телефоне фонарик, засунул его в труднодоступное место, например, под раковину, и показания сами отобразились на экране смартфона. А ещё мастеру управляющей компании было бы очень удобно в подвале или на чердаке снимать показания с прибора учёта, который измеряет потребление ресурсов всем домом сразу.

Задача состоит в разработке модели распознавания показаний индивидуальных и общедомовых приборов учёта.

Данные для построения модели:

1. Архив с фотографиям приборов учёта с описанием: <https://cloud.athene.tech/f/c37eab8f14884b348aef/?dl=1>.
2. Архив с размеченными изображениями - <https://cloud.athene.tech/f/941421353fe1492eaa4c/?dl=1> (дополнено, используйте с умом )

Исходные данные:

1. Файл с заданиями на распознавание показаний ***/mnt/source/task.csv*** со следующими параметрами:
   * *file\_name* - имя файла c изображением.
2. Собственно фотографии приборов учёта для распознавания, находящиеся в той же папке, что и файл с заданиями.

Выходные данные:

1. Файл с распознанными показаниями ***/mnt/result/values.csv*** со следующими параметрами:
   * *file\_name* - имя файла c изображением (такое же, как и в *task.csv*);
   * *data1* - целая часть показаний до запятой;
   * *data2* - дробная часть показаний после запятой.

Примечания:

* Формат всех файлов CSV: с заголовком, разделители - «;».

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.**