ДОКУМЕНТАЦИЯ РАЗРАБОТЧИКА

**vectorization.py**

Данный модуль содержит функции для векторизации данных. Для векторизации используются словари, составленные на основе векторизации исходного набора данных.

**Функции модуля**

|  |  |
| --- | --- |
| **vectorize**(record) | |
| Функция для векторизации одной записи о пользователе (стандартный лог системы Битрикс). | |
| **Параметры**: | **record: *list*** |
| Запись о пользователе. |
| **Возвращает**: | **record\_vec: *DataFrame*** |
| Векторизованная запись о пользователе. |

|  |  |
| --- | --- |
| **saveList**(myList, filename) | |
| Сохраняет список в файл. | |
| **Параметры**: | **myList: *list*** |
| **filename: *string*** |
|  | Название файла, в который будет сохранен список. Файл должен иметь расширение npy. |

|  |  |
| --- | --- |
| **loadList**(filename) | |
| Функция считывает содержимое файла с расширением npy. | |
| **Параметры**: | **filename: *string*** |
| Название файла. Файл должен иметь расширение npy. |
| **Возвращает**: | ***list*** |
| Список данных из файла. |

|  |  |
| --- | --- |
| **ip\_vectorization**(item) | |
| Функция для векторизации IP. | |
| **Параметры**: | **item: *string*** |
| IP пользователя. |
| **Возвращает**: | **ip: *int*** |
| Векторизованное значение IP. |

|  |  |
| --- | --- |
| **url\_vectorization**(item) | |
| Функция для векторизации url. | |
| **Параметры**: | **item: *string*** |
| url. |
| **Возвращает**: | **url: *int*** |
| Векторизованное значение url. |

|  |  |
| --- | --- |
| **user\_agent\_vectorization**(item) | |
| Функция для векторизации User Agent. | |
| **Параметры**: | **item: *string*** |
| User Agent. |
| **Возвращает**: | **platform: *int*** |
| Векторизованное значение операционной системы. |
| **browser: *int*** |
| Векторизованное значение браузера. |

|  |  |
| --- | --- |
| **date\_time\_vectorization**(item) | |
| Функция для векторизации даты и времени. | |
| **Параметры**: | **item: *string*** |
| Время. |
| **Возвращает**: | **date: *int*** |
| Векторизованное значение даты (день недели). |
| **time: *int*** |
| Векторизованное значение времени (час). |

|  |  |
| --- | --- |
| **user\_vectorization**(item) | |
| Функция для векторизации признака "Пользователь" уникальным номером пользователя. | |
| **Параметры**: | **item: *string*** |
| Время. |
| **Возвращает**: | **user: *int*** |
| Векторизованное значение признака "Пользователь". |
| **admin: *int*** |
| 1, если пользователь является администратором и 0 в противном случае. |

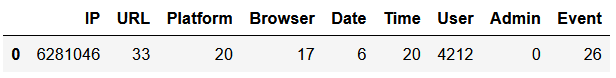
|  |  |
| --- | --- |
| **event\_vectorization**(item) | |
| Функция для векторизации события. | |
| **Параметры**: | **item: *string*** |
| Событие. |
| **Возвращает**: | **event: *int*** |
| Векторизованное значение события. |

**Пример использования**

>>> data\_new = ['2021-01-17 20:52:14', 'вход', '[4212] Иванов Иван Иванович', '95.215.86.111', '/?login', '[4212] Иванов Иван Иванович ', 'nan', 'SECURITY', 'main', 'Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.141 Safari/537.36', 's1']

>>> data\_new\_vec = vectorization.vectorize(data\_new)

>>> data\_new\_vec



**pickle\_classifiers.py**

Данный модуль содержит функции для обучения и сохранения обученных классификаторов.

**Функции модуля**

|  |  |
| --- | --- |
| **creat\_outliers**(d) | |
| Искусственным образом создает выбросы в данных, заменяя значение одного из признаков случайной записи. | |
| **Параметры**: | **d: *DataFrame*** |
| Набор записей. |
| **Возвращает**: | **outliers: *list*** |
| Набор данных, являющихся выбросами. |

|  |  |
| --- | --- |
| **isolation\_forest**(X\_train, user, X\_outliers) | |
| Функция для векторизации одной записи о пользователе (стандартный лог системы Битрикс). | |
| **Параметры**: | **X\_train: *DataFrame*** |
| Тренировочные данные. |
| **user: *int*** |
| Номер пользователя. |
| **X\_outliers: *list*** |
| Выбросы. |
| **Возвращает**: | **clf: *IsolationForest*** |
| Обученный классификатор. |

|  |  |
| --- | --- |
| **save\_clf**(user, clf) | |
| Функция сохраняет обученный классификатор в папку models. | |
| **Параметры**: | **user: *int*** |
| Номер пользователя. |
| **clf: *IsolationForest*** |
| Обученный классификатор. |

|  |  |
| --- | --- |
| **train\_and\_save\_clf**(user, df) | |
| Функция обучает и сохраняет классификатор. | |
| **Параметры**: | **user: *int*** |
| Номер пользователя. |
| **df: *DataFrame*** |
| Набор записей. |

|  |  |
| --- | --- |
| **save\_all\_classifiers**(df) | |
| Функция для обучения и сохранения классификаторов для каждого пользователя из набора данных df. | |
| **Параметры**: | **df: *DataFrame*** |
| Набор записей. |

**Пример использования**

# Загружаем набор данных

data = pd.read\_excel('data.xls')

df = pd.DataFrame(data)

df = df.drop('Unnamed: 0', 1)

# Сохраняем обученные классификаторы для каждого пользователя

save\_all\_classifiers(df)

**API**

Для создания API используется Flask приложение. В коде предусмотрена возможность получать GET запросы.

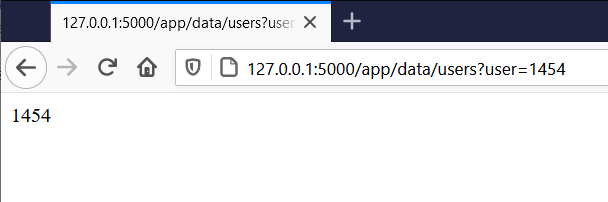
Flask приложение содержит следующие функции:

|  |  |
| --- | --- |
| **home()** | |
| Отвечает на веб-запрос '/'. | |
| **Возвращает**: | ***string*** |
| Ответ HTTP. |

|  |  |
| --- | --- |
| **api\_all()** | |
| Отвечает на веб-запрос '/app/data/users/all'. | |
| **Возвращает**: | ***json*** |
| Информация обо всех пользователях в наборе данных. |

|  |  |
| --- | --- |
| **api\_user()** | |
| Отвечает на веб-запрос '/app/data/users'. | |
| **Возвращает**: | **user: *string*** |
| Значение аргумента user запроса. |
| **Ошибка:** | ***string*** |
| Сообщение об ошибке. |

Пример обращения к запущенному приложению через браузер:



**Docker контейнер**

Для того, чтобы подготовить образ, на базе которого будет собран Docker контейнер, необходимо создать два файла: Dockerfile с описанием команд для запуска написанного ранее приложения, и файл requirements.txt, в котором будут прописаны все необходимые для работы приложения библиотеки.

Содержание файла requirements.txt:

|  |
| --- |
| Flask==1.1.2  numpy==1.19.5  pandas==1.2.0  scikit-learn==0.24.0  scipy==1.6.0  sklearn==0.0  xlrd==2.0.1  xlwt==1.3.0 |

Содержание Dockerfile:

|  |
| --- |
| # установить базовый образ (ОС хоста)  FROM python:3.9  # установка рабочей директории в контейнере  WORKDIR /code  # копирование содержимого файла зависимостей в рабочий каталог  COPY requirements.txt .  # установка зависимостей  RUN pip install -r requirements.txt  # копирование содержимого локального каталога src в рабочий каталог  COPY src/ .  # порт, который будет заниматься Docker контейнером  EXPOSE 5000  # команда, запускающая контейнер  CMD [ "python3", "-m" , "flask", "run", "--host=0.0.0.0"] |

Решение должно иметь следующую структуру:

API

├─── requirements.txt

├─── src

│ └─── app.py

└─── Dockerfile

Для того, чтобы запустить Docker контейнер на вашем сервере, необходимо выполнить следующие шаги.

Установить Docker с официального сайта <https://docs.docker.com/get-docker/>.

Собрать образ можно с помощью команды

docker build -t image\_name .

Для использования одного собранного образа на разных серверах стоит загрузить его в Docker Hub. Для этого image\_name в команде по сборке образа нужно заменить на repository\_name/image\_name, а затем добавит на Docker Hub с помощью интерфейса, предоставляемого установленным ранее Docker Desktop.

Затем запустить контейнер, собранный на базе разработанного образа с помощью команды:

docker run -p 5000:5000 image\_name

Чтобы посмотреть информацию о запущенных контейнерах, используйте команду docker ps, а для остановки контейнера – docker stop container\_name (имя контейнера не совпадает с названием образа; его можно узнать из информации о запущенных контейнерах).