Лабораторная работ 3. Airflow and MLflow.

Грибанов Данил, 6233

Задачи

1. Пайплайн, который обучает любой классификатор из sklearn по заданному набору параметров.
2. Пайплайн, который выбирает лучшую модель из обученных и производит её хостинг.

Ход работы

Все файлы (dockerfile, docker-compose и прочее) находятся в папке req (так же dags и скрипты для запуска соотв. частей).
Важно отметить, что для запуска airflow, используется файл по пути ./req/airflow/docker-compose.yaml
Там находятся модификации для dind и отдельные хранилища для работы (часть из этого осталась еще со 2 лабораторной работы, и больших изменений там нет).
Далее будут описаны кратко основные изменения airflow compose файла.
Перед запуском самой системы (докера airflow) следует подготовить следующее:

Создадим хранилище с данными для запуска и обучения всех систем:

```
volumes:
  postgres-db-volume:
  airflow-data-volume:
    driver: local
    driver_opts:
      type: none
      o: bind
      device: "${AIRFLOW_PROJ_DIR:-.}/data"
```

Добавим данное хранилище с папкой докер-файлов нужных в dind:

```yaml
services:

  docker:
    image: docker:dind      You, 48 minutes ago • Uncommitted changes
    privileged: true
    # Build additional dockers in dind TODO: Is there some way to do init prebuild for dind?
#    command: 'cd /dockerfiles/huggy-face && docker build . -t huggy_face_image'
    environment:
      DOCKER_TLS_CERTDIR: ""
    volumes:
      - airflow-data-volume:/data
      - ${AIRFLOW_PROJ_DIR:-.}/dockerfiles:/dockerfiles

  postgres:
    image: postgres:13
```

Также необходимо добавить переменные для поиска лучшей модели в MLflow:

```yaml
      # See https://airflow.apache.org/docs/docker-stack/entrypo
      DUMB_INIT_SETSID: "0"
      # For MFLOW to search and process best model
      AWS_ACCESS_KEY_ID: ${AWS_ACCESS_KEY_ID}
      AWS_SECRET_ACCESS_KEY: ${AWS_SECRET_ACCESS_KEY}
      MLFLOW_TRACKING_URI: ${MLFLOW_TRACKING_URI}
      MLFLOW_S3_ENDPOINT_URL: ${MLFLOW_S3_ENDPOINT_URL}
    restart: always
    depends_on:
```

Для последующего выполнения заданий, необходимо собрать докер airflow (./req/airflow/docker-compose.yaml) и mlflow docker-compose.mlflow.yaml.

Для докера airflow имеется свой .env с необходимыми параметрами:

```
AIRFLOW_PROJ_DIR=.
AIRFLOW_UID=50000

AWS_ACCESS_KEY_ID=minio
AWS_SECRET_ACCESS_KEY=minio123

#MLFLOW_S3_ENDPOINT_URL=http://minio:9000
MLFLOW_TRACKING_URI=http://mlflow_server:5000
MLFLOW_S3_ENDPOINT_URL=http://minio:9000

MYSQL_DATABASE=mlflow_database
MYSQL_USER=mlflow_user
MYSQL_PASSWORD=mlflow
MYSQL_ROOT_PASSWORD=mysql
```

Для MLflow .env выглядит так:

```
AIRFLOW_PROJ_DIR=.
AIRFLOW_UID=50000

AWS_ACCESS_KEY_ID=minio
AWS_SECRET_ACCESS_KEY=minio123

#MLFLOW_S3_ENDPOINT_URL=http://minio:9000
```

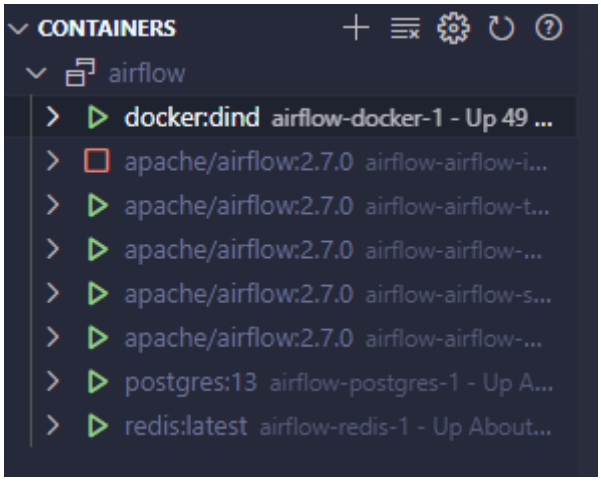
```
MLFLOW_TRACKING_URI=http://localhost:5000
MLFLOW_S3_ENDPOINT_URL=http://localhost:9000

MYSQL_DATABASE=mlflow_database
MYSQL_USER=mlflow_user
MYSQL_PASSWORD=mlflow
MYSQL_ROOT_PASSWORD=mysql
```

Задание 1. Пайплайн, который обучает любой классификатор из sklearn по заданному набору параметров

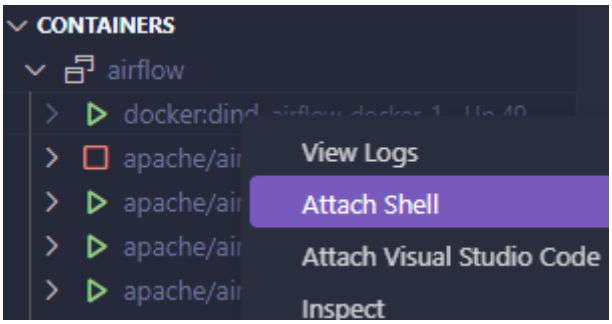
Для работы этого задания нужно сделать пару вещей, а именно:

- Собрать докер внутри dind для обучения
- Сделать соединение в airflow

Начнем с dind. Мы ранее запустили сборку, и имеем подобное:



На всякий случай после запуска дождемся инициализации (пару минут), и зайдем в docker:dind:



Теперь подготовим здесь докер для использования sklearn пакета, обучения и логирования результатов:

cd /dockerfiles/sklearn && docker build . -t sklearn_image

Рассмотрим сам файл докера:

```
FROM python:3.10-slim
# Set where tracing should be sended
ENV MLFLOW_TRACKING_URI=http://mlflow_server:5000
ENV MLFLOW_S3_ENDPOINT_URL=http://minio:9000
ENV AWS_ACCESS_KEY_ID=minio
ENV AWS_SECRET_ACCESS_KEY=minio123
# Set timeout otherwise long build will crash in pip
RUN apt update && apt install -y git
RUN pip install --default-timeout=100 scikit-learn mlflow boto3
```

Здесь нам необходимо проставить путь до MLFLOW и необходимые параметры, а также поставить нужные пакеты. Важно заполнить путь именно таким образом (возможно можно еще заменить mlflow_server на web)

Дождемся сборки. Теперь мы готовы к запуску основного DAG для обучения.

Сам DAG:

```
import os
from datetime import datetime
from airflow import DAG
from airflow.providers.docker.operators.docker import DockerOperator
from airflow.sensors.filesystem import FileSensor

from docker.types import Mount

default_args = {
    'owner': 'airflow',
    'start_date': datetime(2023, 1, 1),
    'retries': 1,
}


dag = DAG(
    'train_nn',
    default_args=default_args,
    description='DAG train NN',
    schedule_interval=None,
)

wait_for_new_file = FileSensor(
    task_id='wait_for_new_train_file',
    poke_interval=10,  # Interval to check for new files (in seconds)
    filepath='/opt/airflow/data/lab3_configs',  # Target folder to monitor
    fs_conn_id='file_train_connection',
    dag=dag,
)

train_nn = DockerOperator(
    task_id='train_nn_on_new_config',
    image='sklearn_image',
    docker_url="tcp://docker:2375", # For Dind usage case
    mount_tmp_dir=False,
    network_mode='host',
#    env_file='/dockerfiles/.env',
```

```
38.     entrypoint='bash',
39.     command=['-c', "python /data/scripts/train_sklearn.py /data/lab3_configs"],
40.     mounts=[
41.         Mount(source='/data', target='/data', type='bind'),
42.     ],
43.     dag=dag,
44. )
45.
46.
47. wait_for_new_file >> train_nn
```

Важно отметить сборку докера для обучения, особенно на 36 строчке с установкой network_mode. Нам необходимо чтобы был доступ к mlflow, значит нам надо прокинуть хост сеть для запускаемого докера, тогда по ранее установленным ссылкам (через .env для airflow-worker и для docker_sklearn) будет доступен mlflow.

Я также пытался здесь настроить через еще один.env, однако почему-то при сборке через DockerOperation он не видит этот файл при сборке, якобы он скрыт. Думал chmod проставить, но как-то не вышло, поэтому решил явно в dockerfile прописать.

Код обучения:

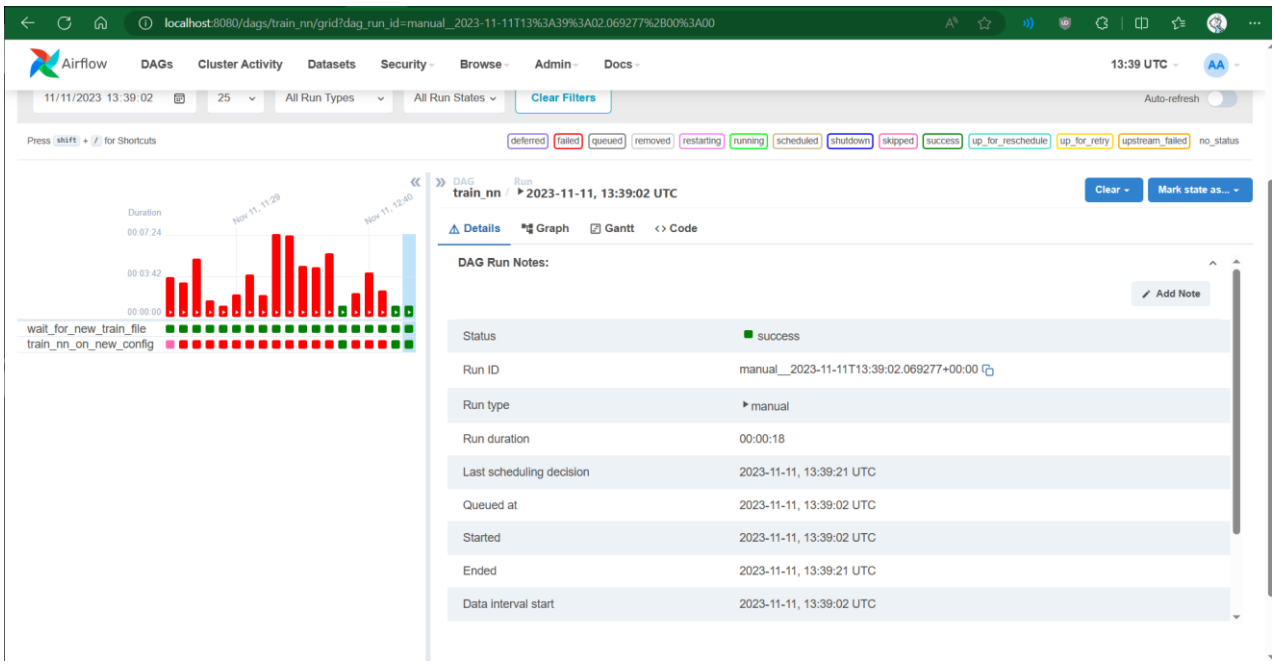
```
1.  import argparse
2.  import glob
3.  import os
4.  import json
5.  import datetime
6.
7.  from sklearn.datasets import load_digits
8.  from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
9.  from sklearn.ensemble import AdaBoostClassifier, RandomForestClassifier
10. from sklearn.gaussian_process import GaussianProcessClassifier
11. from sklearn.gaussian_process.kernels import RBF
12. from sklearn.inspection import DecisionBoundaryDisplay
13. from sklearn.model_selection import train_test_split
14. from sklearn.naive_bayes import GaussianNB
15. from sklearn.neighbors import KNeighborsClassifier
16. from sklearn.neural_network import MLPClassifier
17. from sklearn.pipeline import make_pipeline
18. from sklearn.preprocessing import StandardScaler
19. from sklearn.svm import SVC
20. from sklearn.tree import DecisionTreeClassifier
21. from sklearn.metrics import f1_score, accuracy_score
22.
23. from mlflow.models import infer_signature
24. import mlflow
25.
26. AVAILABLE_CLASSIFIERS_NAME2MODEL_DICT = {
27.     'KNeighborsClassifier': KNeighborsClassifier,
28.     'SVC': SVC,
```

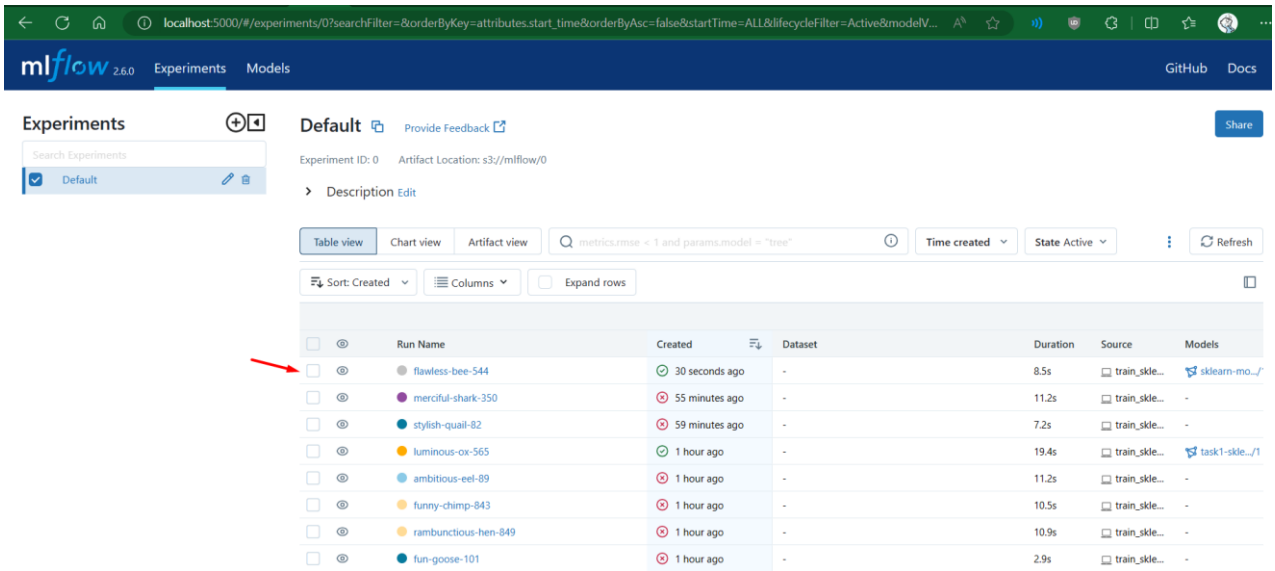
```
    'GaussianProcessClassifier': GaussianProcessClassifier,
    'DecisionTreeClassifier': DecisionTreeClassifier,
    'RandomForestClassifier': RandomForestClassifier,
    'MLPClassifier': MLPClassifier,
    'AdaBoostClassifier': AdaBoostClassifier,
    'GaussianNB': GaussianNB,
    'QuadraticDiscriminantAnalysis': QuadraticDiscriminantAnalysis,
}

def train(config):
    with open(config, 'r') as jr:
        config_data = json.load(jr)

    if AVAILABLE_CLASSIFIERS_NAME2MODEL_DICT.get(config_data['model']) is None:
        raise Exception('Unknown model name')

    with mlflow.start_run():
        digits = load_digits()
        # flatten the images
        n_samples = len(digits.images)
        data = digits.images.reshape((n_samples, -1))
        # Split data into 50% train and 50% test subsets
        X_train, X_test, y_train, y_test = train_test_split(
            data, digits.target, test_size=0.5, shuffle=False
        )


        clf = AVAILABLE_CLASSIFIERS_NAME2MODEL_DICT.get(config_data['model'])(**config_data['parameters'])
        mlflow.log_params(config_data)
        # Learn the digits on the train subset
        clf.fit(X_train, y_train)

        # Predict the value of the digit on the test subset
        predicted = clf.predict(X_test)

        f1_w = f1_score(y_test, predicted, average='weighted')
        f1_avg = f1_score(y_test, predicted, average='macro')
        acc = accuracy_score(y_test, predicted)
        mlflow.log_metric("f1_w", f1_w)
        mlflow.log_metric("f1_avg", f1_avg)
        mlflow.log_metric("acc", acc)

        signature = infer_signature(X_test, predicted)
        mlflow.sklearn.log_model(
            clf, 'sk_models',
            signature=signature,
            # Name must be unique, so just add datetime
            registered_model_name='sklearn-model-%s-%s' % (config_data['model'], datetime.datetime.now()),
        )


if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='Train sklearn model on config file.')
    parser.add_argument('configs_folder_path', type=str,
                        help='Path to folder with configs of the sklearn models.')

    args = parser.parse_args()
    configs_files_path = glob.glob(os.path.join(args.configs_folder_path, '*.json'))
    for config_file_path in configs_files_path:
        train(config_file_path)
```

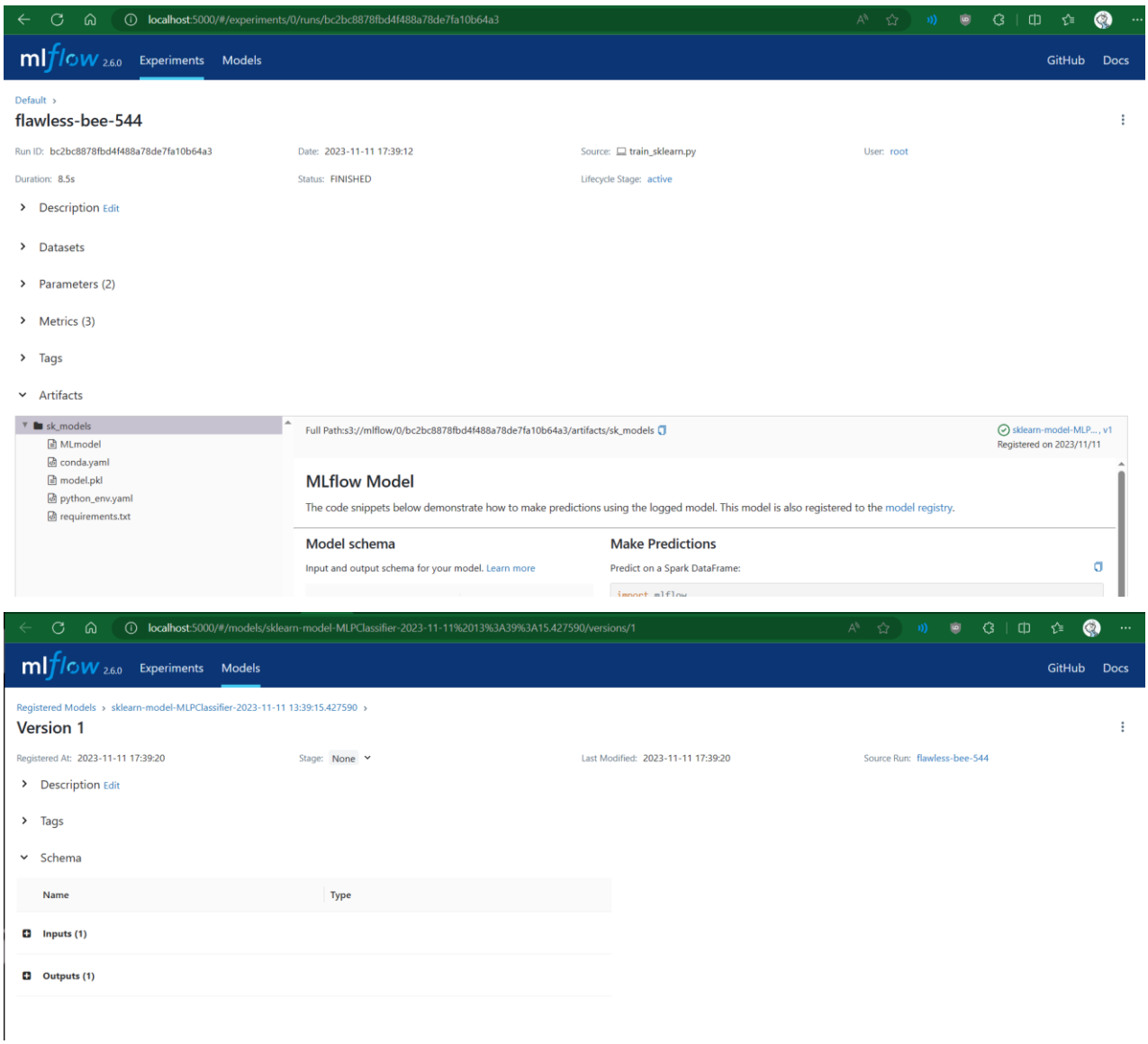
Пример json конфига:

```
{
    "model": "MLPClassifier",
    "parameters": {
        "hidden_layer_sizes": 200,
        "learning_rate": "adaptive"
    }
}
```

Пример запуска:







Задание 2. Пайплайн, который выбирает лучшую модель из обученных и производит её хостинг.

Теперь необходимо чтобы автоматически проверялись модели и лучшей ставился статус продакшена (Production).

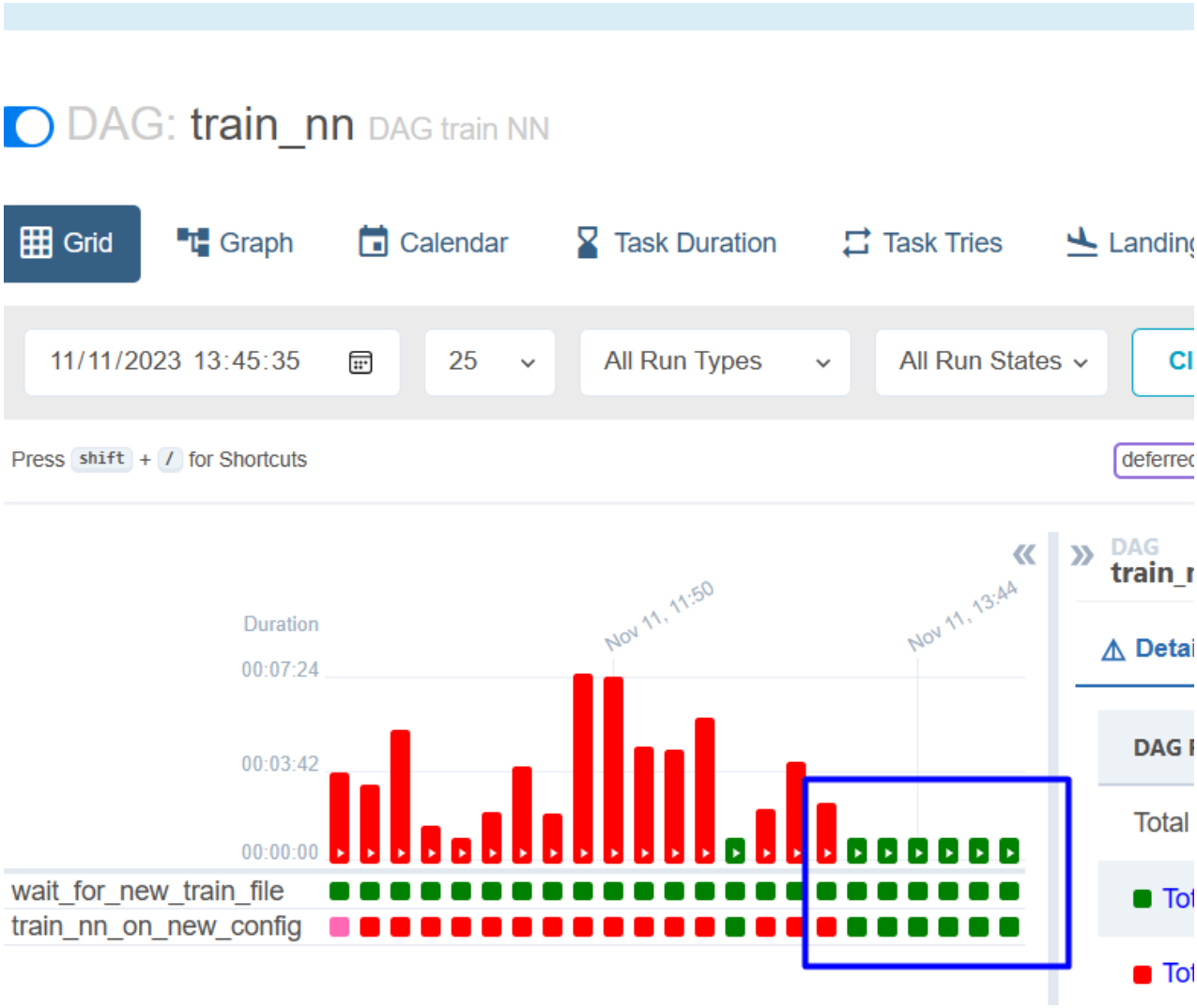
Для этого используется следующий DAG:

```
1.  from datetime import datetime, timedelta
2.  from airflow.sensors.time_delta import TimeDeltaSensorAsync
3.  from airflow.decorators import task, dag
4.
5.
6.  @dag(
7.      dag_id='choose_prod_model',
8.      start_date=datetime(2021, 1, 1),
9.      schedule_interval=None,
10.     catchup=False,
11.     doc_md=__doc__,
12. )
```

```python
def choose_prod_model():
    wait_interval = TimeDeltaSensorAsync(
        task_id="wait_interval",
        delta=timedelta(seconds=10),
    )

    @task
    def check_mlflow_models():
        from mlflow import MlflowClient

        client = MlflowClient()
        best_f1_w = -1.0
        best_model = None

        for rm in client.search_registered_models():
            latest_model_version = rm.latest_versions[-1]
            exp_info = client.get_run(latest_model_version.run_id)

            if exp_info.data.metrics['f1_w'] > best_f1_w:
                best_f1_w = exp_info.data.metrics['f1_w']
                best_model = latest_model_version

        if best_model is not None:
            client.transition_model_version_stage(
                name=best_model.name, version=best_model.version, stage="Production"
            )
        else:
            print('Models not found')

    wait_interval >> check_mlflow_models()

choose_prod_model_pipeline = choose_prod_model()
```

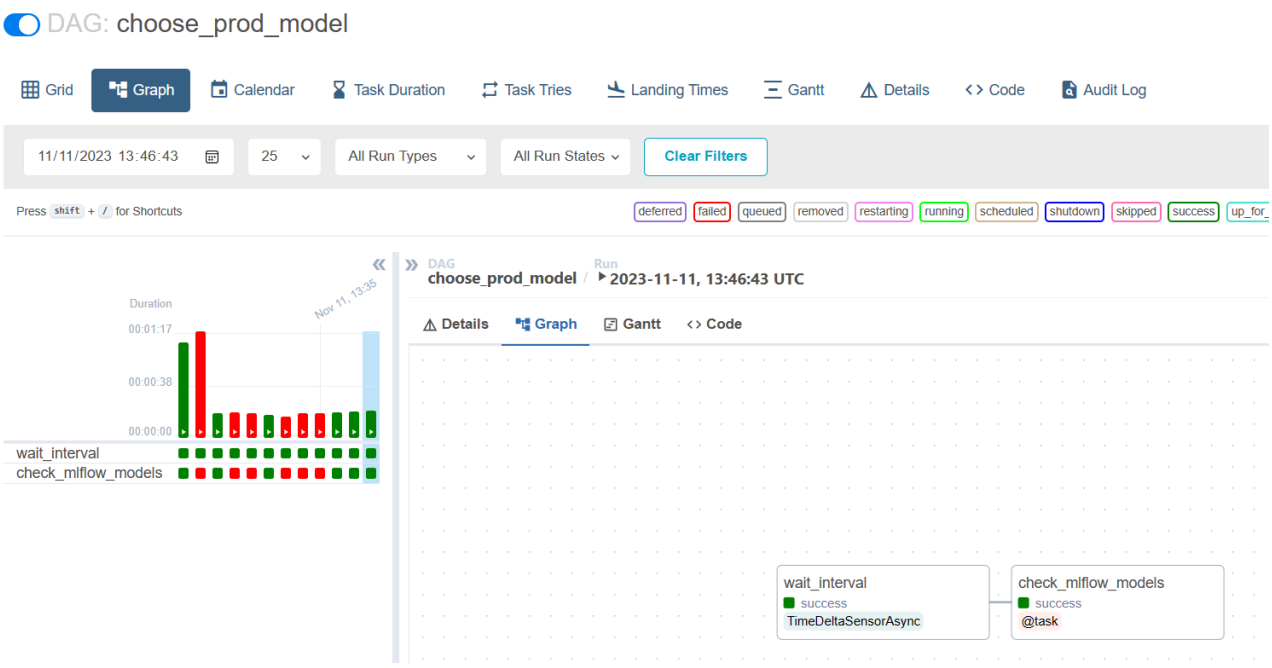
Ранее был запущено обучение и сохранилась модель, запустим подобный скрипт несколько раз и затем запустим DAG описанный выше.

Запускаем несколько раз обучение:

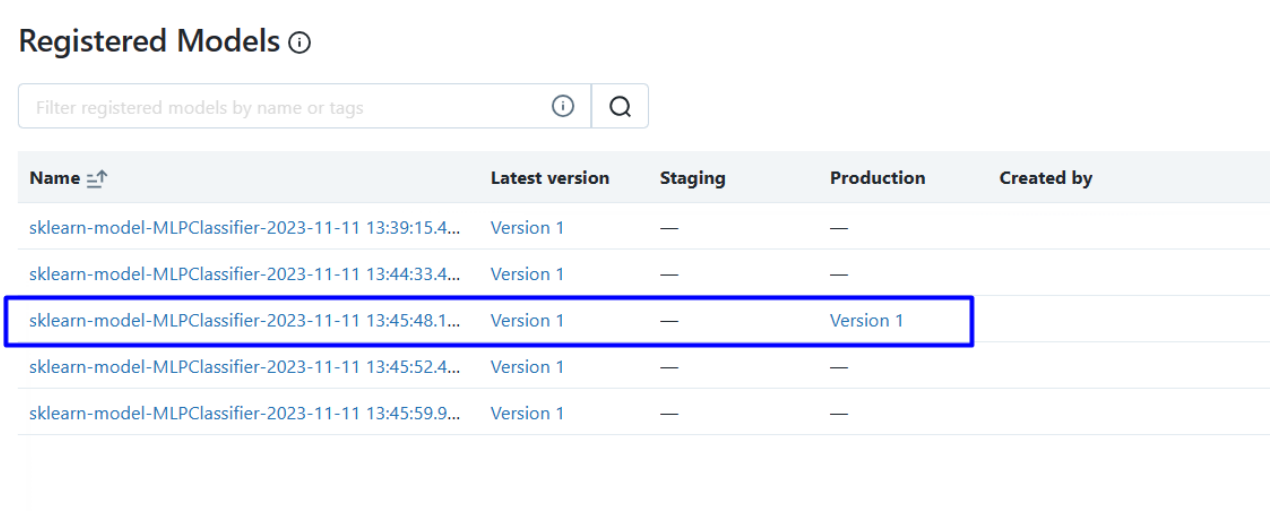


Видим несколько моделей в итоге:

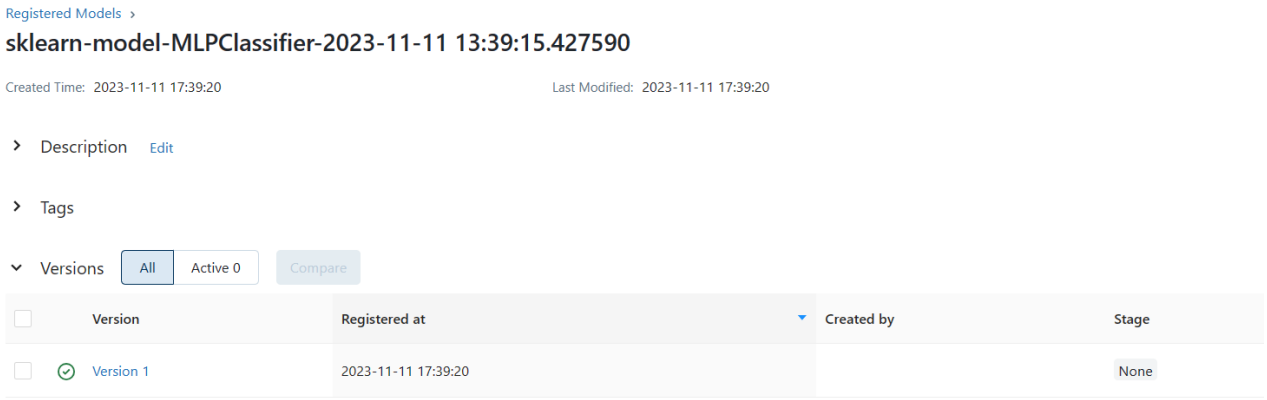
Запускаем DAG для поиска лучшей:



Проверяем:



Другие модели не имеют подобное статуса, пример:



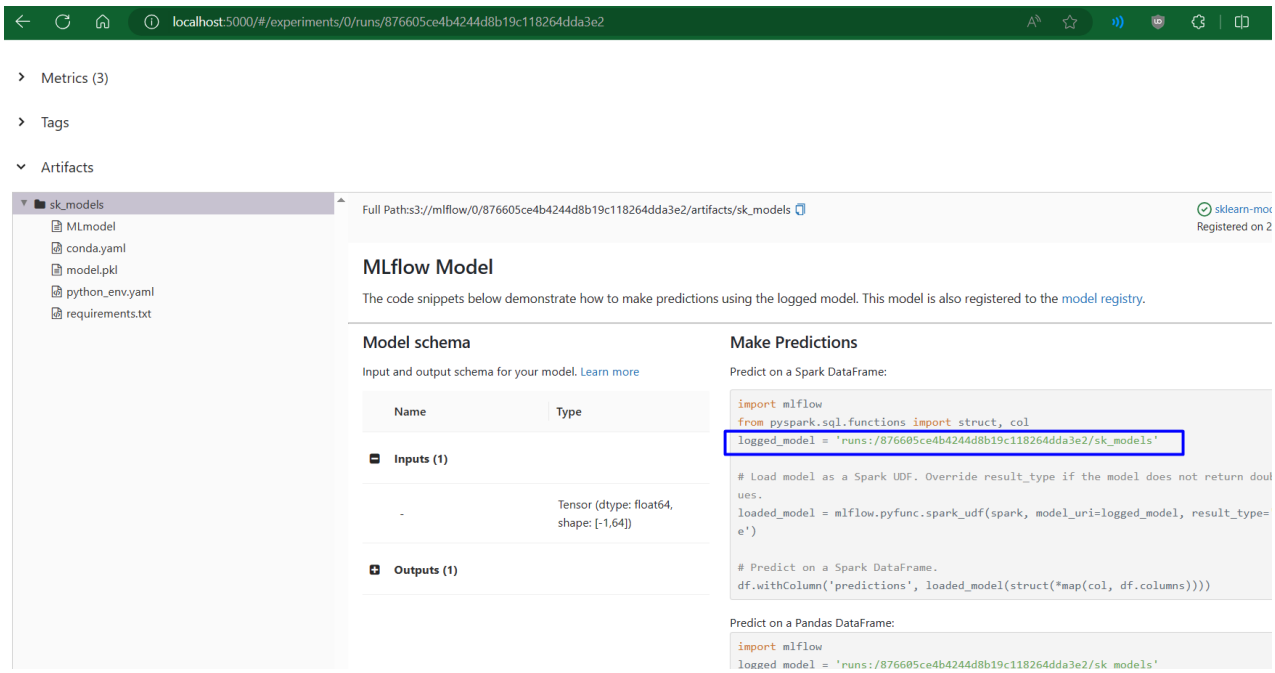
Теперь, данную модель можно захостить через сам MLflow и MLServer. Некоторая помарка, я не делал отдельных скриптов и отдельных вызовов для следующий дествий, так как они довольно скромные и делаются быстро. Я установил все нужное и делал все в консоли в airflow-worker для удобства. Во-первых установим MLFlow с extra пакетами (помним, просто в airflow-worker для удобства и быстроты, далее все там же):

```
pip install mlflow[extras]
```

Теперь необходимо запустить модель с:

```
mlflow models serve -m runs:/876605ce4b4244d8b19c118264dda3e2/sk_models --enable-mlserver --env-manager local
```

В качестве параметра -m указывается model_uri (значение runs:/876605ce4b4244d8b19c118264dda3e2/sk_models), которую можно получить, например отсюда:



Я взял это значение для Production сети.

```
airflow@a85189324515:/opt/airflow$ mlflow models serve -m runs:/876605ce4b4244d8b19c118264dda3e2/sk_models --enable-mlserver --env-manager local
2023/11/11 14:06:00 INFO mlflow.models.flavor_backend_registry: Selected backend for flavor 'python_function'
2023/11/11 14:06:00 WARNING mlflow.pyfunc.mlserver: Timeout is not yet supported in MLServer.
2023/11/11 14:06:00 INFO mlflow.pyfunc.backend: === Running command 'exec mlserver start /tmp/tmpfj9tqd4w/sk_models'
/home/airflow/.local/lib/python3.8/site-packages/starlette_exporter/middleware.py:97: FutureWarning: group_paths and filter_unhandled_paths will ch
e to True in the next release. See https://github.com/stephenhillier/starlette_exporter/issues/79 for more info
  warnings.warn(
2023-11-11 14:06:04,481 [mlserver.parallel] DEBUG - Starting response processing loop...
2023-11-11 14:06:04,485 [mlserver.rest] INFO - HTTP server running on http://127.0.0.1:5000
INFO:     Started server process [1182]
INFO:     Waiting for application startup.
```

```
INFO:     Started server process [1182]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
2023-11-11 14:06:06,625 [mlserver.grpc] INFO - gRPC server running on http://127.0.0.1:8081
INFO:     Application startup complete.
INFO:     Uvicorn running on http://127.0.0.1:5000 (Press CTRL+C to quit)
INFO:     Uvicorn running on http://127.0.0.1:8082 (Press CTRL+C to quit)
2023/11/11 14:06:10 WARNING mlflow.pyfunc: Detected one or more mismatches between the model's dependencies and the current Python environment:
 - numpy (current: 1.24.4, required: numpy==1.26.1)
 - packaging (current: 23.1, required: packaging==23.2)
 - psutil (current: 5.9.5, required: psutil==5.9.6)
 - scipy (current: 1.10.1, required: scipy==1.11.3)
To fix the mismatches, call `mlflow.pyfunc.get_model_dependencies(model_uri)` to fetch the model's environment and install dependencies using the r
esulting environment file.
2023/11/11 14:06:10 WARNING mlflow.pyfunc: The version of Python that the model was saved in, `Python 3.10.13`, differs from the version of Python
that is currently running, `Python 3.8.17`, and may be incompatible
2023-11-11 14:06:11,168 [mlserver] INFO - Loaded model 'mlflow-model' succesfully.
2023-11-11 14:06:11,176 [mlserver] INFO - Loaded model 'mlflow-model' succesfully.
```

Модель готова к работе.

Возьмем следующий вход:

```
In [21]: d.images[0]
Out[21]:
array([[ 0.,  0.,  5., 13.,  9.,  1.,  0.,  0.],
       [ 0.,  0., 13., 15., 10., 15.,  5.,  0.],
       [ 0.,  3., 15.,  2.,  0., 11.,  8.,  0.],
       [ 0.,  4., 12.,  0.,  0.,  8.,  8.,  0.],
       [ 0.,  5.,  8.,  0.,  0.,  9.,  8.,  0.],
       [ 0.,  4., 11.,  0.,  1., 12.,  7.,  0.],
       [ 0.,  2., 14.,  5., 10., 12.,  0.,  0.],
       [ 0.,  0.,  6., 13., 10.,  0.,  0.,  0.]])

In [22]: d.target[0]
Out[22]: 0
```

По очертаниям, это нуль, что подтверждает значение метки (нулевой класс, или нуль имеется в виду цифра из набора данных мини версии «mnist»).

Перепишем входные данные в виде запроса:

```
curl http://127.0.0.1:5000/invocations -H 'Content-Type: application/json' -d '{
  "dataframe_split": {
      "columns": [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
  20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41,
  42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63],
      "data": [[0.0,  0.0,  5.0, 13.0,  9.0,  1.0,  0.0,  0.0, 0.0,  0.0, 13.0, 15.0,
  10.0, 15.0,  5.0,  0.0, 0.0,  3.0, 15.0,  2.0,  0.0, 11.0,  8.0,  0.0, 0.0,  4.0,
  12.0,  0.0,  0.0,  8.0,  8.0,  0.0, 0.0,  5.0,  8.0,  0.0,  0.0,  9.0,  8.0,  0.0,
  0.0,  4.0, 11.0,  0.0,  1.0, 12.0,  7.0,  0.0, 0.0,  2.0, 14.0,  5.0, 10.0,
  12.0,  0.0,  0.0, 0.0,  0.0,  6.0, 13.0, 10.0,  0.0,  0.0,  0.0]]
  }
}'
```

В другой консоли отправим этот запрос:

```
airflow@a85189324515:/opt/airflow$ curl http://127.0.0.1:5000/invocations -H 'Content-Type: application/json' -d '{
>   "dataframe_split": {
>       "columns": [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 3
4, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63],
>       "data": [[0.0,  0.0,  5.0, 13.0,  9.0,  1.0,  0.0,  0.0, 0.0,  0.0, 13.0, 15.0, 10.0, 15.0,  5.0,  0.0, 0.0,  3.0, 15.0,  2.0,  0.0, 11.0,
 8.0,  0.0, 0.0,  4.0, 12.0,  0.0,  0.0,  8.0,  8.0,  0.0, 0.0,  5.0,  8.0,  0.0,  0.0,  9.0,  8.0,  0.0, 0.0,  4.0, 11.0,  0.0,  1.0, 12.0,  7.0,
 0.0, 0.0,  2.0, 14.0,  5.0, 10.0, 12.0,  0.0,  0.0, 0.0,  0.0,  6.0, 13.0, 10.0,  0.0,  0.0,  0.0]]
>   }
> }'
{"predictions": [0]}airflow@a85189324515:/opt/airflow$
```

И видим, что мы получаем ответ класс 0.

Для другого числа (единица):

```
In [26]: d.images[200]
Out[26]:
array([[ 0.,  0.,  0.,  0., 11., 12.,  0.,  0.],
       [ 0.,  0.,  0.,  3., 15., 14.,  0.,  0.],
       [ 0.,  0.,  0., 11., 16., 11.,  0.,  0.],
       [ 0.,  0.,  9., 16., 16., 10.,  0.,  0.],
       [ 0.,  4., 16., 12., 16., 12.,  0.,  0.],
       [ 0.,  3., 10.,  3., 16., 11.,  0.,  0.],
       [ 0.,  0.,  0.,  0., 16., 14.,  0.,  0.],
       [ 0.,  0.,  0.,  0., 11., 11.,  0.,  0.]])

In [27]: d.target[200]
Out[27]: 1
```

Ответ:

```
airflow@a85189324515:/opt/airflow$ curl http://127.0.0.1:5000/invocations -H 'Content-Type: application/json' -d '{
>   "dataframe_split": {
>       "columns": [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 3
4, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63],
>       "data": [[ 0.0,  0.0,  0.0,  0.0, 11.0, 12.0,  0.0,  0.0, 0.0,  0.0,  0.0,  3.0, 15.0, 14.0,  0.0,  0.0, 0.0,  0.0,  0.0, 11.0, 16.0, 11.0,
  0.0,  0.0, 0.0,  0.0,  9.0, 16.0, 16.0, 10.0,  0.0,  0.0, 0.0,  4.0, 16.0, 12.0, 16.0, 12.0,  0.0,  0.0, 0.0,  3.0, 10.0,  3.0, 16.0, 11.0,  0.0,
  0.0, 0.0,  0.0,  0.0,  0.0, 16.0, 14.0,  0.0,  0.0, 0.0,  0.0,  0.0,  0.0, 11.0, 11.0,  0.0,  0.0]]
>   }
> }'
{"predictions": [1]}airflow@a85189324515:/opt/airflow$
```