

Андреев Александр 6233

ЛР сделана вместе с Корноушкиным Ильей 6231

Первый pipeline

Был написан скрипт для первого pipeline.

```
8 import os
9
10 os.environ["AWS_ACCESS_KEY_ID"] = "minio"
11 os.environ["AWS_SECRET_ACCESS_KEY"] = "minio123"
12 os.environ["MLFLOW_S3_ENDPOINT_URL"] = "http://minio:9000"
13
14
15 def branch_func():
16     if (os.path.isfile('/opt/airflow/data/x_train.csv') and os.path.isfile('/opt/airflow/data/y_train.csv')
17         and os.path.isfile('/opt/airflow/data/x_val.csv') and os.path.isfile('/opt/airflow/data/y_val.csv')
18         and os.path.isfile('/opt/airflow/data/x_test.csv') and os.path.isfile('/opt/airflow/data/y_test.csv')):
19         return ignore.task_id
20     else:
21         return prepare_data.task_id
22
23 default_args = {
24     'owner': 'airflow',
25     'retries': 1,
26 }
27
28 dag = DAG(
29     'train_model_pipeline',
30     default_args=default_args,
31     description='',
32     start_date = datetime(2023, 11, 19),
33     schedule_interval=None,
34 )
35
36 branch_op = BranchPythonOperator(
37     task_id='branch_task',
38     python_callable=branch_func,
39     trigger_rule="all_done",
40     dag=dag,
41 )
42
43 prepare_data = BashOperator(
44     task_id="prepare_data",
45     bash_command="python /opt/airflow/data/prepare_data.py",
46     dag=dag
47 )
48
49 wait_file = FileSensor(
50     task_id='wait_file',
51     poke_interval=5,
52     filepath='/opt/airflow/data/conf.json',
53     fs_conn_id='default',
54     dag=dag,
55 )
56
57 ignore = EmptyOperator(
58     task_id='ignore'
59 )
60
61 train_model = BashOperator(
62     task_id="train_model",
63     bash_command="python /opt/airflow/data/train.py",
64     trigger_rule=TriggerRule.NONE_FAILED_MIN_ONE_SUCCESS,
65     dag=dag
66 )
67
68 wait_file >> branch_op
69
70 branch_op >> ignore >> train_model
71 branch_op >> prepare_data >> train_model
```

Рисунок 1 – Код первого dag файла

Также было выполнено дополнительное задание: чтобы скрипт `prepare_data` выполнялся только один раз при запуске `dag`. Но на мой взгляд это не совсем логичное задание и нужно его переформулировать: чтобы скрипт `prepare_data` выполнялся в том случае, если в рабочей директории отсутствуют один, несколько или все файлы необходимые для обучения и тестирования модели. Сделано это было с помощью `BranchPythonOperator`.

В один момент была идея «сделать интересно» с использованием `bash` команд, где одна задача командой `bash` определяла наличие файлов, а другой задачей с помощью `xcom_pull` перехватывала бы то сообщение и уже без функций из `os python` возвращала имя нужной задачи.

```
exists_file = BashOperator(
    task_id="exists_file",
    bash_command="[ -f x_train.csv ] && [ -f y_train.csv ] && [ -f x_val.csv ] && [ -f y_val.csv ] && [ -f x_test.csv ] && [ -f y_test.csv ] && echo True || echo False",
    xcom_push=True,
    dag=dag,
)
```

Рисунок 2 – Код `bash` запроса

```
def branch_func(ti):
    xcom_value = ti.xcom_pull(task_ids='exists_file')
    if xcom_value == True:
        return 'train_model'
    else:
        return ['prepare_data', 'train_model']

branch_op = BranchPythonOperator(
    task_id='branch_task',
    python_callable=branch_func,
    dag=dag,
)
```

Рисунок 3 – Код метода `branch_func`

Опыт был довольно интересный в `bash` среде и код работал, но был выбран в итоге более простой вариант.


```
sasha@DESKTOP-GJ9TOML MINGW64 ~/Desktop/Учеба/DE/Prerequisites/airflow/data (main)
$ [ -f x_train.csv ] && [ -f y_train.csv ] && [ -f x_val.csv ] && [ -f y_val.csv ] && [ -f x_test.csv ] && [ -f y_test.csv ] && echo true || echo false
true

sasha@DESKTOP-GJ9TOML MINGW64 ~/Desktop/Учеба/DE/Prerequisites/airflow/data (main)
$ [ -f x_train.csv ] && [ -f y_train.csv ] && [ -f x_val.csv ] && [ -f y_val.csv ] && [ -f x_test.csv ] && [ -f y_test.csv ] && echo true || echo false
false

sasha@DESKTOP-GJ9TOML MINGW64 ~/Desktop/Учеба/DE/Prerequisites/airflow/data (main)
$
```

Рисунок 4 – Запросы в `bash` среде

По итогу работы были получены модели в `mlflow`.

Default  [Provide Feedback](#) 

Experiment ID: 0 Artifact Location: s3://mlflow/0

> Description [Edit](#)

Table

Chart

Evaluation

Experimental













	Run Name
	 MLPClassifier
	 KNeighborsClassifier
	 SVC
 	 RandomForestClassifier
	 SGDClassifier

Рисунок 5 – Запущенные модели

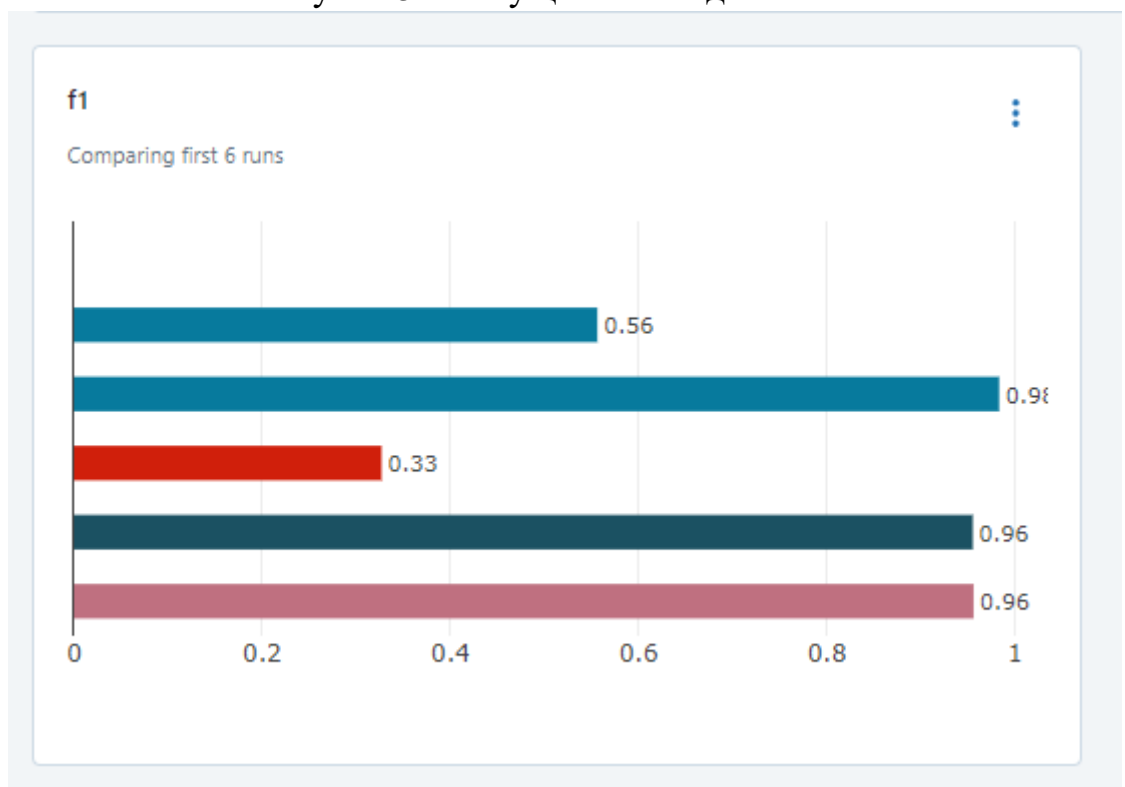


Рисунок 6 – Результаты обучения моделей по метрике f1

Второй pipeline

Аналогично был написан второй dag файл, в котором определялась лучшая модель и выводилась в Production.

```

C: > Users > sasha > Desktop > Учеба > DE > Prerequisites > airflow > dags > validate_model_pipeline.py
1  from datetime import datetime
2  from airflow import DAG
3  from airflow.operators.bash_operator import BashOperator
4  import os
5
6  os.environ["AWS_ACCESS_KEY_ID"] = "minio"
7  os.environ["AWS_SECRET_ACCESS_KEY"] = "minio123"
8  os.environ["MLFLOW_S3_ENDPOINT_URL"] = "http://minio:9000"
9
10 default_args = {
11     'owner': 'airflow',
12     'start_date': datetime(2023, 11, 19),
13     'retries': 1,
14 }
15
16 dag = DAG(
17     'validate_model_pipeline',
18     default_args=default_args,
19     description='',
20     schedule_interval='@daily',
21 )
22
23 validate_model = BashOperator(
24     task_id="validate_model",
25     bash_command="python /opt/airflow/data/validate.py",
26     dag=dag
27 )
28
29 validate_model

```

Рисунок 7 – Код второго dag файла

Name	Latest version	Staging	Production	Created by	Last modified	Tags
KNeighborsClassifier	Version 9	—	Version 1		2023-11-19 15:22:12	—
MLPClassifier	Version 9	—	—		2023-11-19 15:00:23	—
RandomForestClassifier	Version 9	—	—		2023-11-19 15:00:06	—
SGDClassifier	Version 9	—	—		2023-11-19 15:00:03	—
SVC	Version 9	—	—		2023-11-19 15:00:08	—

Рисунок 8 – Запущенная модель