МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное
образовательное учреждение высшего образования
«Самарский национальный исследовательский университет
имени академика С.П. Королева»
(Самарский университет)

Институт информатики и кибернетики
Кафедра суперкомпьютеров и общей информатики

Отчет по лабораторной работе №3

Дисциплина: «Инженерия данных»

Тема: **«Airflow и MLflow – логгирование экспериментов и версионирование моделей»**

Выполнил: Неженский М.С.

Группа: 6233-010402D

Самара 2023

1. Пайплайн для обучения классификаторов

В рамках данной лабораторной работы предлагается построить два пайплайна. Первый – для обучения классификаторов. Для начала стоит создать конфигурационный файл. Я назвал его config.json, он написал на JavaScript. В качестве классификатор были выбраны:

- `AdaBoostClassifier;`
- `RandomForestClassifier;`
- `GaussianProcessClassifier;`
- `KNeighborsClassifier;`
- `MLPClassifier.`

Пример кода:

```
{
        "module": "sklearn.ensemble",
        "classificator": "AdaBoostClassifier",
        "args": {
            "n_estimators": 50,
            "learning_rate": 2.0
        }
}
```

В module записывается библиотека, из которой берем классификатор. В classificator записывается название самого классификатора. Далее в args записываются некоторые параметры классификатора, которые выбираем не по умолчанию.

Например, в данном случае выбраны самостоятельно максимальное количество оценок и размер шага на каждой итерации.

Таким образом создаем файл config.json с пятью выбранными классификаторами.

Код DAG для первого задания с комментариями:

airflow > dags > dag_train_data.py > ...

```python
from datetime import datetime
from airflow import DAG
from airflow.sensors.filesystem import FileSensor
from airflow.operators.bash_operator import BashOperator
import os
os.environ["AWS_ACCESS_KEY_ID"] = "minio"   # переменные среды для сохранения модели
os.environ["AWS_SECRET_ACCESS_KEY"] = "minio123"
os.environ["MLFLOW_S3_ENDPOINT_URL"] = "http://minio:9000"

default_args = { # аргументы по умолчанию
    'owner': 'airflow', # владелец (можно сортировать в airflow)
    'start_date': datetime(2023, 1, 1), # начало работы
    'retries': 1, # количество повторных попыток, которое должно быть выполнено, прежде чем задача будет провалена
}

dag = DAG(
    'train_models', # название DAG
    default_args = default_args, # определение аргументов по умолчанию
    description = 'DAG, which is needed to train models from sklearn', # описание DAG
    schedule_interval = None, # автоматический запуск не планируется, запуск "извне"
)

monitoring = FileSensor( # используется для обнаружения файлов в локальной директории( в данном случае для обнаружения файла config.json)
    task_id = 'monitoring', # название задания
    poke_interval = 20, #  время, в течение которого задание должно ждать между каждой попыткой
    filepath = '/opt/airflow/data/config.json', # путь по которому происходит поиск файла и название файла
    fs_conn_id = 'connection_inference', # название подключения (использовано, что и в прошлой ЛР)
    dag = dag, # использование DAG
)

train_data = BashOperator(
    task_id = "train_data", # название задания
    bash_command = "python /opt/airflow/data/train_data.py", # запуск файла python из данной директории
    dag = dag # использование DAG
)

monitoring >> train_data # последовательность заданий для выполнения, выполняются поочередно
```

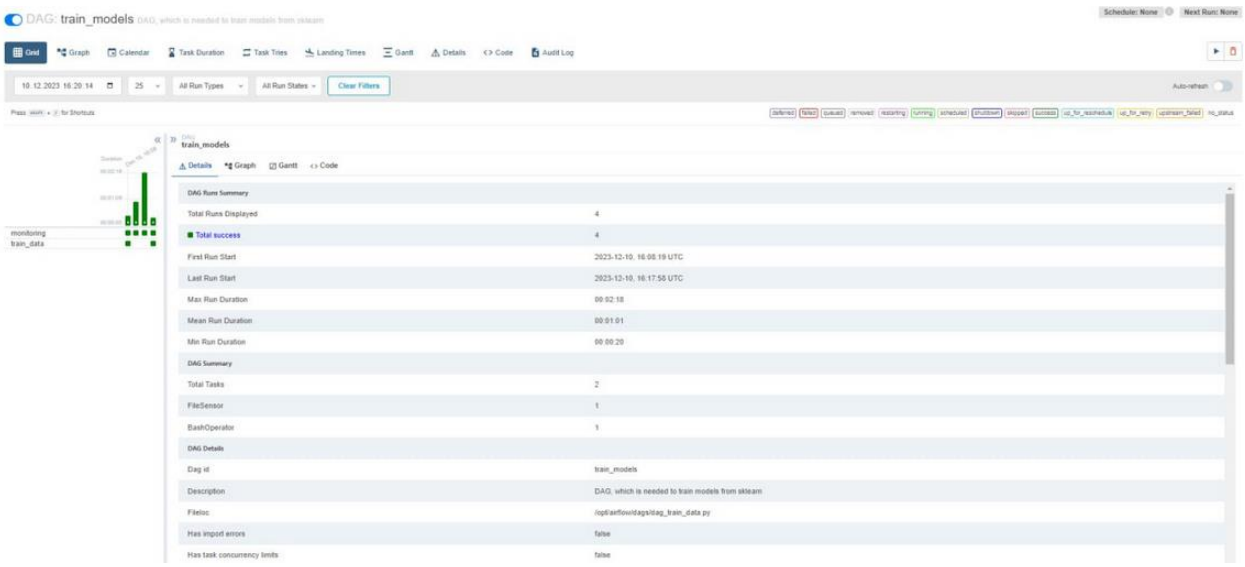
Код на python, содержащий загрузку данных, деление выборки, запуск эксперимента для первого задания с комментариями:

airflow > data > train_data.py > ...

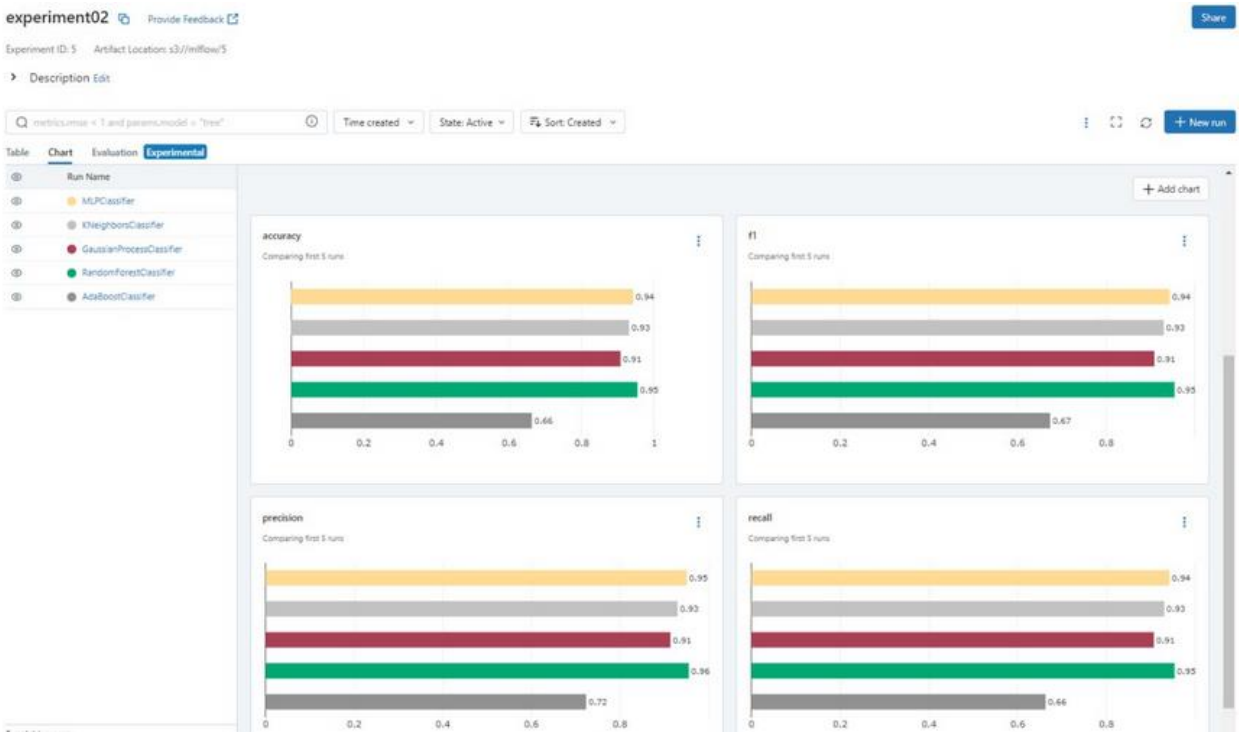
```python
import json
import importlib
import pandas as pd
import numpy as np
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.datasets import load_breast_cancer
import mlflow
import mlflow.sklearn
from mlflow.models import infer_signature

with open('/opt/airflow/data/config.json', 'r') as config_file: # открытие файла с конфигурацией классификаторов
    config_data = json.load(config_file) # загрузка данных о классификаторах в переменную

breast_cancer = load_breast_cancer() # набор данных о раке молочной железы в штате Висконсин (классификация)

X_train, X_test, y_train, y_test = train_test_split( # разделение данных на тестовую и тренировочную выборку
    breast_cancer.data, breast_cancer.target, test_size = 0.3, random_state = 33 # 70 к 30
)
X_validation, X_test, y_validation, y_test = train_test_split( # разделение данных на тестовую и валидационную выборку
    X_test, y_test, test_size = 0.5, random_state = 33 # 50 к 50
)

pd.DataFrame(X_validation).to_csv('/opt/airflow/data/X_validation.csv', index=False) # запись в файл для второго задания
pd.DataFrame(y_validation).to_csv('/opt/airflow/data/y_validation.csv', index=False) # валидационной выборки

tracking_url = 'http://mlflow_server:5000' # задание ссылки для работы с MLflow
mlflow.set_tracking_uri(tracking_url) # подключение к серверу для работы с MLflow

experiment = mlflow.create_experiment("experiment02") # создание эксперимента
mlflow.set_experiment(experiment_id = experiment) # использование созданного эксперимента
```

```python
for i, config in enumerate(config_data['configs']): # для каждого классификатора из конфигурационного файла
    mlflow.start_run(run_name = config['classificator'], experiment_id = experiment) # запуск в эксперименте
    module = importlib.import_module(config['module']) # импортирование для обработки данных в конфигурационном файле
    classificator = getattr(module, config['classificator']) # название классификатора из данных
    model = classificator(**config['args']) # задание модели обучения через аргументы классификатора конфигурационного файла
    model.fit(X_train, y_train) # обучение модели
    prediction = model.predict(X_test) # предсказание модели
    signature = infer_signature(X_test, prediction) # сигнатура

    mlflow.log_params(config['args']) # логгирование параметров модели
    mlflow.log_metrics({"accuracy": accuracy_score(y_test, prediction), # логгирование метрик
                        "precision": precision_score(y_test, prediction, average = 'weighted'),
                        "recall": recall_score(y_test, prediction, average = 'weighted'),
                        "f1": f1_score(y_test, prediction, average = 'weighted')
                        })

    modelInfo = mlflow.sklearn.log_model( # логгирование модели
        sk_model = model, # использованная модель для логгирования
        artifact_path = config['module'], # модуль с помощью которого распознается модель
        signature = signature, # сигнатура
        registered_model_name = config['classificator']) # название модели

    dataFrame = pd.DataFrame({
        "name":config['classificator'], # запись назнания модели
        "uri":modelInfo.model_uri # запись информации о модели
        },
        index=[i])
    dataFrame.to_csv('/opt/airflow/data/models.csv', mode = 'a', header = False) # запись моделей в файл
    mlflow.end_run() # завершение работы
```

Выполненный Dag:



Экран MLflow со значениями метрик для каждого классификатора:



2. Пайплайн для выбора лучшей модели

В данном задании выполняется все аналогично первому, DAG помещается в папку dags, а вспомогательный файл (validation.py) в папку data. Все это конечно в папке airflow. Код с комментариями приведен ниже. Код для DAG:

airflow > dags > dag_validation.py > ...

```python
from datetime import datetime
from airflow import DAG
from airflow.operators.bash_operator import BashOperator
import os

os.environ["AWS_ACCESS_KEY_ID"] = "minio" # переменные среды для сохранения модели
os.environ["AWS_SECRET_ACCESS_KEY"] = "minio123"
os.environ["MLFLOW_S3_ENDPOINT_URL"] = "http://minio:9000"

default_args = { # аргументы по умолчанию
    'owner': 'airflow', # владелец (можно сортировать в airflow)
    'start_date': datetime(2023, 12, 10), # начало работы
    'retries': 1, # количество повторных попыток, которое должно быть выполнено, прежде чем задача будет провалена
}

dag = DAG(
    'validation', # название DAG
    default_args=default_args, # определение аргументов по умолчанию
    description='DAG for validate', # описание DAG
    schedule_interval='@daily', # запуск один раз в день
)

validation = BashOperator(
    task_id="validation", # название задания
    bash_command="python /opt/airflow/data/validation.py", # запуск файла python из данной директории
    dag=dag # использование DAG
)

validation # последовательность заданий для выполнения, выполняются поочередно
```

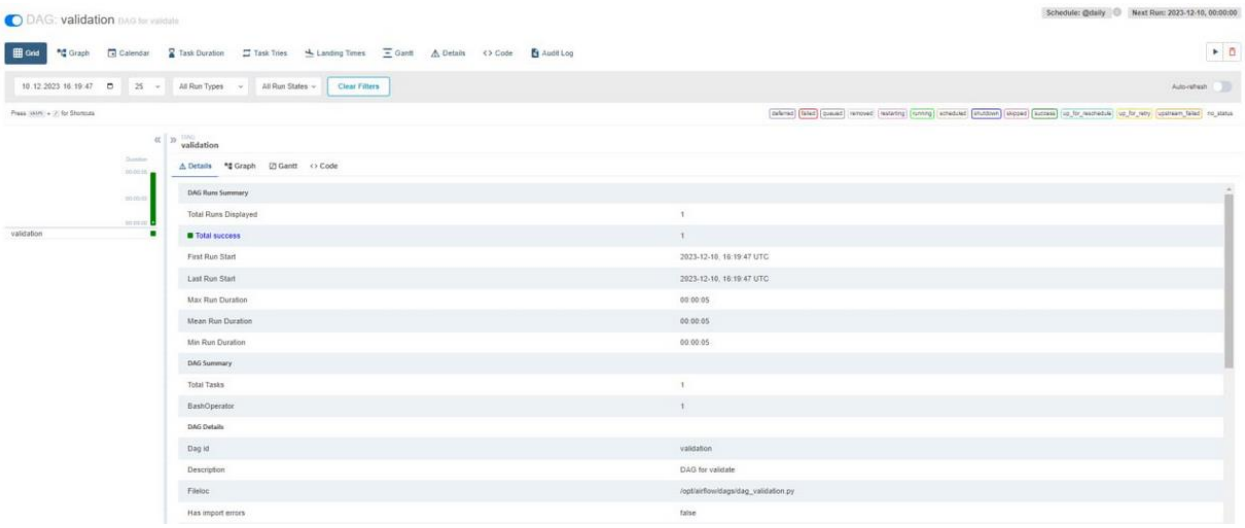
Код validation.py с комментариями:

airflow > data > validation.py > ...

```python
import os
import operator
import pandas as pd
import numpy as np
from sklearn import datasets
# from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score

import mlflow
import mlflow.sklearn
from mlflow import MlflowClient

tracking_uri = 'http://mlflow_server:5000' # задание ссылки для работы с MLflow
mlflow.set_tracking_uri(tracking_uri) # подключение к серверу для работы с MLflow
mlflow.set_experiment("experiment02") # использование созданного эксперимента

X_validation = np.asarray(pd.read_csv(f"/opt/airflow/data/X_validation.csv"), dtype = np.float32) # считывание данных из файла валидационной выборки
y_validation = pd.read_csv(f"/opt/airflow/data/y_validation.csv")

list_models = {} # задание пустого объекта(модели)
mlflow.start_run(run_name = "Production model") # запуск в эксперименте Production model
models = pd.read_csv("/opt/airflow/data/models.csv", header = None) # считывание моделей
for model_Info in models.iterrows(): # для каждой модели считывается записанное название и информация о модели
    name_model = model_Info[1][1]
    uri_model = model_Info[1][2]
    list_models[name_model + " " + uri_model] = mlflow.sklearn.load_model(uri_model) # запись моделей
```

```python
results = {} # задание пустого объекта(результат)
for i, j in list_models.items(): # посчет метрики f1 для каждой модели
    pred = j.predict(X_validation)
    results[i] = f1_score(
        y_validation,
        pred,
        average="weighted"
        )

best_model = max(results, key=results.get) # из всех моделей выбирается та, в которой метрика f1 максимальна
# запись лучшей версии и переопределение stage у лучшей версии с None на Production
version = MlflowClient().search_model_versions(f"name = '{best_model.split(' ')[0]}' and run_id = '{best_model.split(' ')[1].split('/')[1]}'")[0].version
MlflowClient().transition_model_version_stage(name = best_model.split(' ')[0], version = version, stage = "Production")
mlflow.end_run() # завершение работы
```

Выполненный DAG:



Зарегистрированные модели, как видим лучше всего в классификацией справился RandomForestClassifier:

ЗАКЛЮЧЕНИЕ

В результате выполнения лабораторной работы были получены навыки по работе с Docker, Apache Airflow, MLflow. В данной работе были построены два пайплайна – для обучения классификаторов и выбора лучшей модели.