Robert Pavlik

D602 Deployment

QBN1 – Task 3: Program Deployment

GitLab – Task3_Working_Branch

https://gitlab.com/wgu-gitlab-environment/student-repos/rpavli5/d602-deployment-task-3.git

This project aimed to deploy a machine learning model as an API, using FastAPI for the backend and Docker for containerization. The API was designed to accept GET requests and return predictions for average departure delays. This report outlines the development process, the challenges encountered, and the solutions implemented.

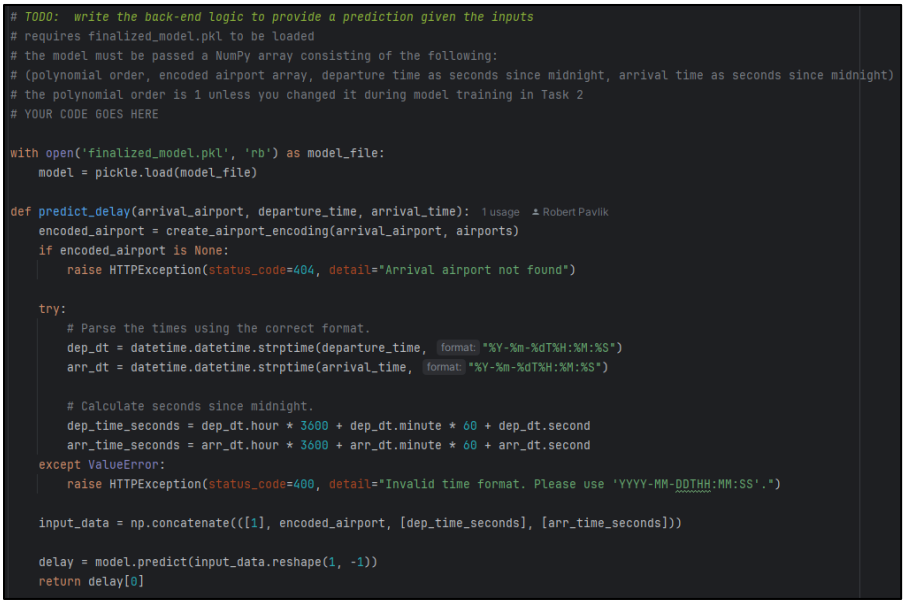
API Development Process

- Technology Choice:
  - FastAPI was used for its modern features, speed, and ease of use in building RESTful APIs.
- Endpoints Implemented: Accepts three query parameters:
  - 'arrival_airport'
  - 'departure_time' (formatted as "YYYY-MM-DDTHH:MM:SS")
  - 'arrival_time' (formatted as "YYYY-MM-DDTHH:MM:SS")

  This endpoint processes the input, applies necessary data transformations, and returns a prediction (average departure delay).

- Model Integration:

  The machine learning model saved as 'finalized_model.pkl', was loaded using Python's pickle module. The API constructs the input array from:
  - A fixed polynomial order (1)
  - A one-hot encoded vector for the arrival airport
  - The departure and arrival times converted to seconds since midnight



```
# TODO:  write the back-end logic to provide a prediction given the inputs
# requires finalized_model.pkl to be loaded
# the model must be passed a NumPy array consisting of the following:
# (polynomial order, encoded airport array, departure time as seconds since midnight, arrival time as seconds since midnight)
# the polynomial order is 1 unless you changed it during model training in Task 2
# YOUR CODE GOES HERE

with open('finalized_model.pkl', 'rb') as model_file:
    model = pickle.load(model_file)

def predict_delay(arrival_airport, departure_time, arrival_time):  1 usage   Robert Pavlik
    encoded_airport = create_airport_encoding(arrival_airport, airports)
    if encoded_airport is None:
        raise HTTPException(status_code=404, detail="Arrival airport not found")

    try:
        # Parse the times using the correct format.
        dep_dt = datetime.datetime.strptime(departure_time, format: "%Y-%m-%dT%H:%M:%S")
        arr_dt = datetime.datetime.strptime(arrival_time, format: "%Y-%m-%dT%H:%M:%S")

        # Calculate seconds since midnight.
        dep_time_seconds = dep_dt.hour * 3600 + dep_dt.minute * 60 + dep_dt.second
        arr_time_seconds = arr_dt.hour * 3600 + arr_dt.minute * 60 + arr_dt.second
    except ValueError:
        raise HTTPException(status_code=400, detail="Invalid time format. Please use 'YYYY-MM-DDTHH:MM:SS'.")

    input_data = np.concatenate(([1], encoded_airport, [dep_time_seconds], [arr_time_seconds]))

    delay = model.predict(input_data.reshape(1, -1))
    return delay[0]
```

Challenges and Resolutions

- Time Conversion Issue:
  - Challenge:
    - Initially, the code converted the provided timestamps to seconds since Jan 1, 1900. This resulted in very large, unrealistic numbers being fed into the model
  - Resolution:
    - I updated the time conversion logic to compute the number of seconds since midnight. This was achieved by parsing the timestamp and calculating

```
# Parse the times using the correct format.
dep_dt = datetime.datetime.strptime(departure_time, format: "%Y-%m-%dT%H:%M:%S")
arr_dt = datetime.datetime.strptime(arrival_time, format: "%Y-%m-%dT%H:%M:%S")

# Calculate seconds since midnight.
dep_time_seconds = dep_dt.hour * 3600 + dep_dt.minute * 60 + dep_dt.second
arr_time_seconds = arr_dt.hour * 3600 + arr_dt.minute * 60 + arr_dt.second
```

      This change aligned the input with the model training parameters and produced more plausible prediction values.

- Dependency Management:
  - Challenge:
    - During testing, the environment was missing some modules, such as FastAPI, causing import errors.
  - Resolution:
    - I ensured that all dependencies were explicitly listed in the ‘requirements.txt’ file and installed using

      ‘ python -m pip install -r requirements.txt ‘

      I also verified the active Python environment to avoid conflicts.

- CI/CD Pipeline Configuration:
  - Challenge:
    - The Gitlab CI pipeline initially failed because the Python image was not specified in the test stage.
  - Resolution:
    - I modified the ‘.gitlab-ci.yml’ file to use the ‘python: 3.12’ image for the pytest stage, ensuring that the CI environment matched the project requirements.

Dockerization

- Dockerfile Creation:
  - I created a Dockerfile based on the ‘python: 3.12.1-slim-bookworm’ image. The Dockerfile copies the project files, installs dependencies, and runs the API using Uvicorn. This containerization facilitates consistent deployment across different environments.
- Pipeline Integration:
  - The Dockerfile, along with the updated CI configuration, allows the GitLab pipeline to build a Docker image automatically. This image is then used to deploy the live API, ensuring that all changes are tracked, and the container runs reliably.

Conclusion

The project demonstrated the successful deployment of a machine-learning model as an API. Key Achievements include:

- Developing a FastAPI-based API with robust endpoints
- Implementing unit tests to validate both successful and error scenarios
- Overcoming challenges in time data conversion, dependency management, and CI/CD configuration.
- Containerizing the application with Docker and integrating it into a GitLab CI/CD pipeline.

Predictions_API.py

```python
# TODO:  write the back-end logic to provide a prediction given the inputs
# requires finalized_model.pkl to be loaded
# the model must be passed a NumPy array consisting of the following:
# (polynomial order, encoded airport array, departure time as seconds since midnight, arrival time as seconds since midnight)
# the polynomial order is 1 unless you changed it during model training in Task 2
# YOUR CODE GOES HERE

with open('finalized_model.pkl', 'rb') as model_file:
    model = pickle.load(model_file)

def predict_delay(arrival_airport, departure_time, arrival_time):  1 usage   Robert Pavlik
    encoded_airport = create_airport_encoding(arrival_airport, airports)
    if encoded_airport is None:
        raise HTTPException(status_code=404, detail="Arrival airport not found")

    try:
        # Parse the times using the correct format.
        dep_dt = datetime.datetime.strptime(departure_time,  format: "%Y-%m-%dT%H:%M:%S")
        arr_dt = datetime.datetime.strptime(arrival_time,  format: "%Y-%m-%dT%H:%M:%S")

        # Calculate seconds since midnight.
        dep_time_seconds = dep_dt.hour * 3600 + dep_dt.minute * 60 + dep_dt.second
        arr_time_seconds = arr_dt.hour * 3600 + arr_dt.minute * 60 + arr_dt.second
    except ValueError:
        raise HTTPException(status_code=400, detail="Invalid time format. Please use 'YYYY-MM-DDTHH:MM:SS'.")

    input_data = np.concatenate(([1], encoded_airport, [dep_time_seconds], [arr_time_seconds]))

    delay = model.predict(input_data.reshape(1, -1))
    return delay[0]
```

```python
# TODO:  write the API endpoints.
# YOUR CODE GOES HERE

#Initializing FastAPI app
/
app = FastAPI()

#Root endpoint to check if the API is functional
/
@app.get("/")  Robert Pavlik
async def root():
    return {"message": "API is functional!"}

#Prediction endpoint to get the average departure delay
/predict/delays
@app.get("/predict/delays")  Robert Pavlik
async def predict_delays(arrival_airport: str, departure_time: str, arrival_time: str):
    try:
        delay = predict_delay(arrival_airport, departure_time, arrival_time)
        return {"average_departure_delay": delay}
    except HTTPException as e:
        raise e
```

Test_predictions.py

```python
#!/usr/bin/env python
# coding: utf-8

from fastapi.testclient import TestClient
from prediction_api import app


client = TestClient(app)

def test_root():  Robert Pavlik
    """Testing the root endpoint"""
    response = client.get("/")
    assert response.status_code == 200
    assert response.json() == {"message": "API is functional!"}

def test_invalid_airport():  Robert Pavlik
    """Testing the /predict/delays endpoint (with an invalid airport)"""
    response = client.get( url: "/predict/delays", params={
        "arrival_airport": "XYZ",  # Invalid airport code
        "departure_time": "2024-10-31T14:30:00",
        "arrival_time": "2024-10-31T22:15:00"
    })
    assert response.status_code == 404
    assert response.json() == {"detail": "Arrival airport not found"}

def test_invalid_time_format():  Robert Pavlik
    """Testing the /predict/delays endpoint (with an invalid time format)"""
    response = client.get( url: "/predict/delays", params={
        "arrival_airport": "JFK",
        "departure_time": "14:30",  # Invalid format
        "arrival_time": "2024-10-31T22:15:00"
    })
    assert response.status_code == 400
    assert response.json() == {"detail": "Invalid time format. Please use 'YYYY-MM-DDTHH:MM:SS'."}
```

PyTest – Run

```
(base) PS C:\Users\pavli\Desktop\WGU COURSE\D602_Task_3> pytest
============================= test session starts =============================
platform win32 -- Python 3.12.7, pytest-7.4.4, pluggy-1.3.0
rootdir: C:\Users\pavli\Desktop\WGU COURSE\D602_Task_3
plugins: anyio-3.7.1
collected 3 items

test_predictions.py ...                                                  [100%]

============================== warnings summary ===============================
..\..\..\anaconda3\lib\site-packages\httpx\_client.py:680
  C:\Users\pavli\anaconda3\lib\site-packages\httpx\_client.py:680: DeprecationWarning: The 'app' shortcut is now deprecated. Use the explicit style 'transport=WSGITransport(app=...)' instead.
    warnings.warn(message, DeprecationWarning)

-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
======================== 3 passed, 1 warning in 1.33s =========================
(base) PS C:\Users\pavli\Desktop\WGU COURSE\D602_Task_3>
```

Dockerfile

```
# Use the Python 3.12.1 slim image based on Debian Bookworm.
FROM python:3.12.1-slim-bookworm

# Set the working directory in the container.
WORKDIR /app

# Copy the requirements file into the container.
COPY requirements.txt .

# Install dependencies.
RUN pip install --no-cache-dir -r requirements.txt

# Copy the rest of the application code.
COPY . .

# Expose port 8000 for the API.
EXPOSE 8000

# Run the API with Uvicorn.
CMD ["uvicorn", "prediction_api:app", "--host", "0.0.0.0", "--port", "8000"]
```

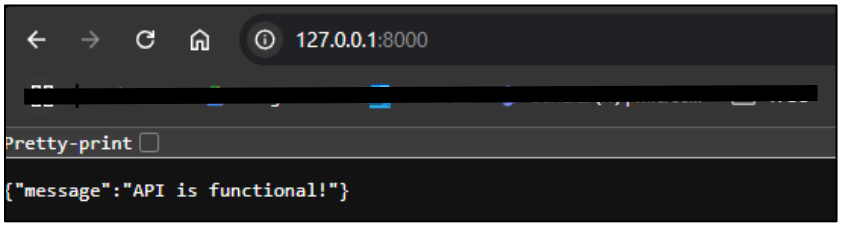
Docker Build

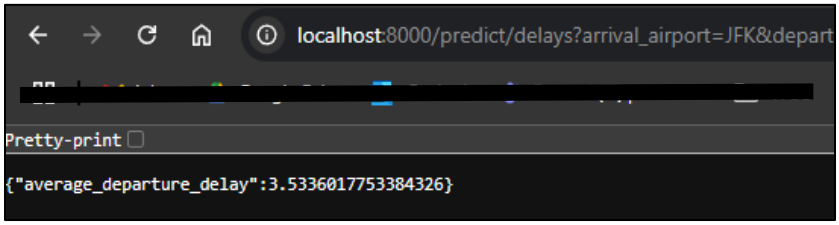
```
(base) PS C:\Users\pavli\Desktop\WGU COURSE\D602_Task_3> docker build -t myapi .
[+] Building 18.5s (11/11) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 563B
 => [internal] load metadata for docker.io/library/python:3.12.1-slim-bookworm
 => [auth] library/python:pull token for registry-1.docker.io
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/5] FROM docker.io/library/python:3.12.1-slim-bookworm@sha256:a64ac5be6928c6a94f00b16e09cdf3ba3edd44452d10ffa4516a58004873573e
 => => resolve docker.io/library/python:3.12.1-slim-bookworm@sha256:a64ac5be6928c6a94f00b16e09cdf3ba3edd44452d10ffa4516a58004873573e
 => [internal] load build context
 => => transferring context: 308.28MB
 => CACHED [2/5] WORKDIR /app
 => CACHED [3/5] COPY requirements.txt .
 => CACHED [4/5] RUN pip install --no-cache-dir -r requirements.txt
 => [5/5] COPY . .
 => exporting to image
 => => exporting layers
 => => exporting manifest sha256:fad06c45dcbc95771276b4b53935f22e6dce4e84e5004f4354d58f1f48f12e41
 => => exporting config sha256:6f03d5abc98ccb547865af23fb6f6b4e57a8b6dee9927e893522e98650136339
 => => exporting attestation manifest sha256:62ab8057a48990af2d3dac30e38ce9d225a1e8bab439c1c6b785c3437fa9c48d
 => => exporting manifest list sha256:36cc22606d88afd15680e10ff179084701e36d120506b71d02a7152170f44077
 => => naming to docker.io/library/myapi:latest
 => => unpacking to docker.io/library/myapi:latest
(base) PS C:\Users\pavli\Desktop\WGU COURSE\D602_Task_3>
```

Docker Running

```
(base) PS C:\Users\pavli\Desktop\WGU COURSE\D602_Task_3> docker run -p 8000:8000 myapi
INFO:     Started server process [1]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
```

```
(base) PS C:\Users\pavli\Desktop\WGU COURSE\D602_Task_3> docker run -p 8000:8000 myapi
INFO:     Started server process [1]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
INFO:     172.17.0.1:51492 - "GET / HTTP/1.1" 200 OK
INFO:     172.17.0.1:52818 - "GET /predict/delays?arrival_airport=JFK&departure_time=2024-10-31T14:30:00&arrival_time=2024-10-31T22:15:00 HTTP/1.1" 200 OK
```






```
(base) PS C:\Users\pavli\Desktop\WGU COURSE\D602_Task_3> curl "http://localhost:8000/predict/delays?arrival_airport=JFK&departure_time=2024-10-31T14:30:00&arrival_time=2024-10-31T22:15:00"


StatusCode        : 200
StatusDescription : OK
Content           : {"average_departure_delay":3.5336017753384326}
RawContent        : HTTP/1.1 200 OK
                    Content-Length: 46
                    Content-Type: application/json
                    Date: Sun, 23 Mar 2025 18:42:12 GMT
                    Server: uvicorn

                    {"average_departure_delay":3.5336017753384326}
Forms             : {}
Headers           : {[Content-Length, 46], [Content-Type, application/json], [Date, Sun, 23 Mar 2025 18:42:12 GMT], [Server, uvicorn]}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 46


(base) PS C:\Users\pavli\Desktop\WGU COURSE\D602_Task_3>
```

```
(base) PS C:\Users\pavli\Desktop\WGU COURSE\D602_Task_3> curl "http://localhost:8000/predict/delays?arrival_airport=JFK&departure_time=14:30&arrival_time=2024-10-31T22:15:00"
curl : {"detail":"Invalid time format. Please use 'YYYY-MM-DDTHH:MM:SS'."}
At line:1 char:1
+ curl "http://localhost:8000/predict/delays?arrival_airport=JFK&depart ...
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : InvalidOperation: (System.Net.HttpWebRequest:HttpWebRequest) [Invoke-WebRequest], WebException
    + FullyQualifiedErrorId : WebCmdletWebResponseException,Microsoft.PowerShell.Commands.InvokeWebRequestCommand
(base) PS C:\Users\pavli\Desktop\WGU COURSE\D602_Task_3>
```

GitLab CI/CD Pipeline