Phase 6

1 Phase 6: Model Scalability and Deployment

course: Machine Learning Algorithms (MAAI).

Student Name: Mina Ezach Naeem Faltos

Student Number: 34388

1.1 A. Introduction

At this last phase we move the Champion Model (Random Forest v1.1) from an enviroment for static training to a live, production-ready RESTful API, production RESTful API. This process satisfy the project requirement for **Model Deployment** to make the model available for external use through a web service.

1.2 B. Phase Objectives

This phase deals with final evaluation criteria by showing: 1. **Model Persistence**: Production use of the trained model artifact (.pkl). 2. **Deployment Implementation**: Initiate a FastApi server, which provides real-time predictions. 3. **Strong Data Handling**: Using an industry grade sanitizer to process raw inputs, and deal with missing features. 4. **Operational Metrics**: Latency (ms) and Model Versioning are used to monitor the performance of the system.

[1]:
```
# Step 1: Resetting dependency and environment sync
import sys
print("Removing conflicting libraries.....")
%pip uninstall -y multipart python-multipart

print("Installing production-stable dependencies.....")
# Preventing unpickling warnings by aligning scikit-learn version
%pip install python-multipart==0.0.9 pandas fastapi uvicorn nest_asyncio joblib␣
↪scikit-learn==1.8.0 openpyxl

print("\n ENVIRONMENT READY. Please restart the kernel to apply these changes.")
```

```
Removing conflicting libraries…
Found existing installation: python-multipart 0.0.9
Uninstalling python-multipart-0.0.9:
  Successfully uninstalled python-multipart-0.0.9
```

```
Note: you may need to restart the kernel to use updated packages.
Installing production-stable dependencies…

WARNING: Skipping multipart as it is not installed.

Collecting python-multipart==0.0.9
  Using cached python_multipart-0.0.9-py3-none-any.whl.metadata (2.5 kB)
Requirement already satisfied: pandas in c:\users\mina\anaconda3\lib\site-
packages (2.3.3)
Requirement already satisfied: fastapi in c:\users\mina\anaconda3\lib\site-
packages (0.128.0)
Requirement already satisfied: uvicorn in c:\users\mina\anaconda3\lib\site-
packages (0.40.0)
Requirement already satisfied: nest_asyncio in c:\users\mina\anaconda3\lib\site-
packages (1.6.0)
Requirement already satisfied: joblib in c:\users\mina\anaconda3\lib\site-
packages (1.5.3)
Requirement already satisfied: scikit-learn==1.8.0 in
c:\users\mina\anaconda3\lib\site-packages (1.8.0)
Requirement already satisfied: openpyxl in c:\users\mina\anaconda3\lib\site-
packages (3.1.5)
Requirement already satisfied: numpy>=1.24.1 in
c:\users\mina\anaconda3\lib\site-packages (from scikit-learn==1.8.0) (2.4.0)
Requirement already satisfied: scipy>=1.10.0 in
c:\users\mina\anaconda3\lib\site-packages (from scikit-learn==1.8.0) (1.16.3)
Requirement already satisfied: threadpoolctl>=3.2.0 in
c:\users\mina\anaconda3\lib\site-packages (from scikit-learn==1.8.0) (3.6.0)
Requirement already satisfied: python-dateutil>=2.8.2 in
c:\users\mina\anaconda3\lib\site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\mina\anaconda3\lib\site-
packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in
c:\users\mina\anaconda3\lib\site-packages (from pandas) (2025.3)
Requirement already satisfied: starlette<0.51.0,>=0.40.0 in
c:\users\mina\anaconda3\lib\site-packages (from fastapi) (0.50.0)
Requirement already satisfied: pydantic>=2.7.0 in
c:\users\mina\anaconda3\lib\site-packages (from fastapi) (2.12.5)
Requirement already satisfied: typing-extensions>=4.8.0 in
c:\users\mina\anaconda3\lib\site-packages (from fastapi) (4.15.0)
Requirement already satisfied: annotated-doc>=0.0.2 in
c:\users\mina\anaconda3\lib\site-packages (from fastapi) (0.0.4)
Requirement already satisfied: anyio<5,>=3.6.2 in
c:\users\mina\anaconda3\lib\site-packages (from
starlette<0.51.0,>=0.40.0->fastapi) (4.12.1)
Requirement already satisfied: idna>=2.8 in c:\users\mina\anaconda3\lib\site-
packages (from anyio<5,>=3.6.2->starlette<0.51.0,>=0.40.0->fastapi) (3.11)
Requirement already satisfied: click>=7.0 in c:\users\mina\anaconda3\lib\site-
packages (from uvicorn) (8.3.1)
Requirement already satisfied: h11>=0.8 in c:\users\mina\anaconda3\lib\site-
```

```
packages (from uvicorn) (0.16.0)
Requirement already satisfied: et-xmlfile in c:\users\mina\anaconda3\lib\site-
packages (from openpyxl) (2.0.0)
Requirement already satisfied: colorama in c:\users\mina\anaconda3\lib\site-
packages (from click>=7.0->uvicorn) (0.4.6)
Requirement already satisfied: annotated-types>=0.6.0 in
c:\users\mina\anaconda3\lib\site-packages (from pydantic>=2.7.0->fastapi)
(0.7.0)
Requirement already satisfied: pydantic-core==2.41.5 in
c:\users\mina\anaconda3\lib\site-packages (from pydantic>=2.7.0->fastapi)
(2.41.5)
Requirement already satisfied: typing-inspection>=0.4.2 in
c:\users\mina\anaconda3\lib\site-packages (from pydantic>=2.7.0->fastapi)
(0.4.2)
Requirement already satisfied: six>=1.5 in c:\users\mina\anaconda3\lib\site-
packages (from python-dateutil>=2.8.2->pandas) (1.17.0)
Using cached python_multipart-0.0.9-py3-none-any.whl (22 kB)
Installing collected packages: python-multipart
Successfully installed python-multipart-0.0.9
Note: you may need to restart the kernel to use updated packages.

 ENVIRONMENT READY. Please restart the kernel to apply these changes.
```

This cell re-saves the model in the current environment to ensure 100% stability and resolve version mismatch warnings.

1.2.1 Please note that a Kerenel restart might be needed to ensure `scikit-learn 1.8.0` is loaded correctly, preventing version mismatch errors during model loading.

[2]:
```
# Step 2: Model persistence and sync

import pandas as pd
import joblib
from sklearn.ensemble import RandomForestRegressor

# 1. Loading Training Data
train_df = pd.read_csv('work_MLA_phase2_34388_train.csv')
train_df = train_df[train_df['price'] <= 500]
train_df['is_private_room'] = train_df['room_type'].apply(lambda x: 1 if x ==␣
 ↪'Private room' else 0)

# 2. Defining the features
features = ['accommodates', 'bathrooms', 'bedrooms', 'beds',␣
 ↪'review_scores_rating', 'latitude', 'longitude', 'is_private_room']
X = train_df[features]
y = train_df['price']
```

```
# 3. Training the Champion Model (v1.1)
sync_model = RandomForestRegressor(n_estimators=100, max_depth=20,␣
 ↪random_state=42)
sync_model.fit(X, y)

# 4. Saving Artifact
joblib.dump(sync_model, 'airbnb_price_model.pkl')
print("Model airbnb_price_model.pkl synchronized and saved for deployment.")
```

```
Model airbnb_price_model.pkl synchronized and saved for deployment.
```

[ ]:
```
# Step 3: The final deployment

import uvicorn
from fastapi import FastAPI, UploadFile, File, HTTPException
from fastapi.openapi.docs import get_swagger_ui_html
from fastapi.responses import HTMLResponse, JSONResponse
import pandas as pd
import numpy as np
import nest_asyncio
import joblib
import os, time, gc, asyncio, json
from io import BytesIO
from contextlib import asynccontextmanager

# 1. Configuring
MODEL_PATH = "airbnb_price_model.pkl"
MODEL_VERSION = "v1.1_2026_01"
model_artifact = None

@asynccontextmanager
async def lifespan(app: FastAPI):
    global model_artifact
    if os.path.exists(MODEL_PATH):
        model_artifact = joblib.load(MODEL_PATH)
        print("INFO: Real Model Loaded Successfully.")
    yield
    gc.collect()

app = FastAPI(docs_url=None, redoc_url=None, lifespan=lifespan)

# 2. Industrial sanitizer
def industrial_sanitize_data(df):
    """
    Handles raw files by mapping synonyms, stripping currency symbols ($),
    and generating missing columns with safe defaults.
    """
    df.columns = df.columns.str.lower().str.strip()

    # Mapping Synonyms
    rename_map = {
        'lat': 'latitude', 'lng': 'longitude', 'guests': 'accommodates',
        'rating': 'review_scores_rating', 'bath': 'bathrooms', 'bed': 'beds'
    }
    df = df.rename(columns=rename_map)

    # Engineering Feature
    if 'room_type' in df.columns:
        df['is_private_room'] = np.where(df['room_type'].astype(str).str.
↪lower().str.contains('private'), 1, 0)

    required = ['accommodates', 'bathrooms', 'bedrooms', 'beds',␣
↪'review_scores_rating', 'latitude', 'longitude', 'is_private_room']
    defaults = {'accommodates': 2, 'bathrooms': 1, 'bedrooms': 1, 'beds': 1,␣
↪'review_scores_rating': 90, 'latitude': 41.38, 'longitude': 2.17,␣
↪'is_private_room': 0}

    for col in required:
        if col not in df.columns:
            df[col] = defaults[col] # Automatically creating the missing column

        # Cleaning currency strings
        if df[col].dtype == 'object':
            df[col] = df[col].astype(str).str.replace('$', '', regex=False).str.
↪replace(',', '', regex=False)

        # Forcing numeric, coerce errors to NaN, filling with default
        df[col] = pd.to_numeric(df[col], errors='coerce').fillna(defaults[col])

    return df[required]

# 3. UI Setup
@app.get("/docs", include_in_schema=False)
async def custom_swagger_ui_html():
    html = get_swagger_ui_html(openapi_url=app.openapi_url, title="Airbnb␣
↪Pricing Intelligence",
                               swagger_ui_parameters={"docExpansion": "full",␣
↪"defaultModelsExpandDepth": -1})
    custom_code = """
    <style>
        body { font-family: -apple-system, sans-serif; background-color:␣
↪#f8f9fa; }
```

```
        .swagger-ui .wrapper { max-width: 800px !important; margin: 0 auto !
↪important; padding-top: 50px !important; }
        .swagger-ui .topbar, .swagger-ui .scheme-container { display: none !
↪important; }
        .swagger-ui .info { text-align: center !important; }
        .swagger-ui .info .title { font-size: 36px !important; font-weight: 700␣
↪!important; }
        .swagger-ui .opblock { background: white !important; border: none !
↪important; border-radius: 12px !important; box-shadow: 0 4px 12px␣
↪rgba(0,0,0,0.1) !important; padding: 20px !important; }
        .swagger-ui .btn.execute { background-color: #0d6efd !important; color:␣
↪white !important; height: 50px !important; font-size: 18px; width: 48% !
↪important; border-radius: 6px; }
        .swagger-ui .btn.clear { background-color: #6c757d !important; color:␣
↪white !important; height: 50px !important; font-size: 18px; width: 48% !
↪important; border-radius: 6px; margin-left: 4% !important; }
        .swagger-ui .curl-command, .swagger-ui .request-url, .swagger-ui .
↪prop-type, .swagger-ui .opblock-section-header { display: none !important; }
    </style>
    <script>window.onload = function() { setTimeout(function() { document.
↪querySelector('.try-out__btn')?.click(); }, 500); }</script>
    """
    return HTMLResponse(html.body.decode("utf-8").replace("</head>",␣
↪custom_code + "</head>"))

# 4. Endpoint for prediction
@app.post("/upload_file_for_prediction", summary="Upload Excel File for␣
↪Analysis")
async def upload_file(file: UploadFile = File(...)):
    start_time = time.time()
    try:
        contents = await file.read()
        df_input = pd.read_excel(BytesIO(contents)) if file.filename.endswith('.
↪xlsx') else pd.read_csv(BytesIO(contents), low_memory=False)
        X_pred = industrial_sanitize_data(df_input.copy())

        # Batch Prediction
        preds = model_artifact.predict(X_pred)

        # Strict sanitization: eliminate nans/inf for json compliance
        preds = np.nan_to_num(preds, nan=0.0, posinf=0.0, neginf=0.0)

        df_input['Predicted_Price_EUR'] = np.round(preds, 2)
        df_input['Model_Version'] = MODEL_VERSION
        df_input['Latency_ms'] = round((time.time() - start_time) * 1000, 2)
        # Strict Conversion for Dictionary (Replaceing DataFrame NaNs with 0 or␣
↪None)
        clean_results = df_input.head(200).replace([np.inf, -np.inf], np.nan).
↪fillna(0).to_dict(orient="records")

        return JSONResponse(content=clean_results)
    except Exception as e:
        return JSONResponse(status_code=500, content={"error": str(e)})

# 5. Server start
if __name__ == "__main__":
    nest_asyncio.apply()
    loop = asyncio.get_event_loop()
    config = uvicorn.Config(app, host="127.0.0.1", port=8024, loop="asyncio")
    server = uvicorn.Server(config)

    # Checking if loop is already running
    if not loop.is_running():
        loop.run_until_complete(server.serve())
    else:
        loop.create_task(server.serve())

    print("Website Live: http://127.0.0.1:8024/docs")
```

```
Website Live: http://127.0.0.1:8024/docs

INFO:     Started server process [11356]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://127.0.0.1:8024 (Press CTRL+C to quit)

INFO: Real Model Loaded Successfully.
INFO:     127.0.0.1:1464 - "GET /docs HTTP/1.1" 200 OK
INFO:     127.0.0.1:1464 - "GET /openapi.json HTTP/1.1" 200 OK
```

1.3 C. Verification of the system and Conclusion.

This system creates a live dashboard that is available at the address of `http://127.0.0.1:8024/docs`.

Verification: 1. **JSON Compliance**: The logic of nan to num makes sure that the responses are valid removing 500 Errors. 2. **Industrial Robustness**: The industrial_sanitize_data engine successfully handles the input of the new data by generating the missing feature and repairing currency strings. 3. **Stability**: The "asyncio" task handler prevents "RuntimeError" in Jupyter environments.