Lab 5 - Containerize and Deploy the Model with Streamlit App

Write the Dockerfile for FastAPI App by adding it to root of the source code.

```
.
├── Dockerfile
├── src
│   ├── api
│   │   ├── README.md
│   │   ├── inference.py
│   │   ├── main.py
│   │   ├── requirements.txt
│   │   ├── schemas.py
│   │   └── utils.py
```

File: `house-price-predictor/Dockerfile`

```
FROM python:3.11-slim

WORKDIR /app

COPY src/api/ .

RUN pip install -r requirements.txt

COPY models/trained/*.pkl models/trained/

EXPOSE 8000

CMD [ "uvicorn",  "main:app",  "--host",  "0.0.0.0",  "--port",  "8000" ]
```

Build image with

```
docker image build -t fastapi .
```

where, replace `xxxxxx` with actual DockerHub username.

Validate

```
docker image ls
docker image history fastapi
```

Test run it

```
docker run -idt -p 8888:8000 --name api fastapi
```

Validate

```
docker ps -n 1
docker logs api
```

You could validate by accessing it on http://localhost:8888/docs

once done, you could remove the test container as

```
docker rm -f api
```

Packaging Streamlit App

Switch to `streamlit_app` path

```
cd streamlit_app
```

Create a Dockerfile with the following content:

File : `streamlit_app/Dockerfile`

```
FROM python:3.9-slim

WORKDIR /app

COPY  app.py requirements.txt .

RUN pip install -r requirements.txt

EXPOSE 8501

CMD  [ "streamlit", "run", "app.py", "--server.address=0.0.0.0" ]
```

This time we will build the image with a tag that you could publish to the registry later

```
[replace xxxx with your docker id]
docker image build -t xxxx/streamlit:v1 .
```

validate

```
docker image ls
```

run

```
docker run -idt -p 8501:8501 --name web xxxx/streamlit:v1
```

validate

```
docker ps -n 1
docker logs web
```

Access it using http://localhost:8501

You will see the streamlit based web ui, however its not connected to the model yet, which is alright.

Once done testing, remove the streamlit container as

```
docker rm -f web
```

Packaging Model Serving Infra with Docker Compose

Create a Docker Compose spec with both `fastapi` and `streamlit` app as,

File : `house-price-predictor/docker-compose.yaml`

```
services:
  fastapi:
    image: xxxx/fastapi:dev   #replace with actual docker user id
    build:
      context: .
      dockerfile: Dockerfile
    ports:
      - 8000:8000

  streamlit:
    image: xxxx/streamlit:dev  #replace with actual docker user id
    build:
      context: ./streamlit_app
      dockerfile: Dockerfile
    ports:
```

```
      - 8501:8501
    environment:
      API_URL: http://fastapi:8000
```

Build images with

```
docker compose build
```

validate

```
docker image ls
```

bring it up as a stack of services as

```
docker compose up -d
```

Common Error

If you see an error such as

```
Error: unable to start container
"16848c53ba2019f76bacd8f0427a47fe24de4fcf0360e1d07f1a4c35c09f3188": cannot
listen on the TCP port: listen tcp4 :8000: bind: address already in use
Error: something went wrong with the request: "listen tcp :8501: bind: address
already in use\n"
```

This happens because other container/service is listening on this port. In such case change the left side of prt mapping in compose spec

```
    ports:
```

```
      - 8005:8000
```

where it would listen to and be available at port `8005` instead of previous `8000`

Once the compose stack is up, you should be able to validate with

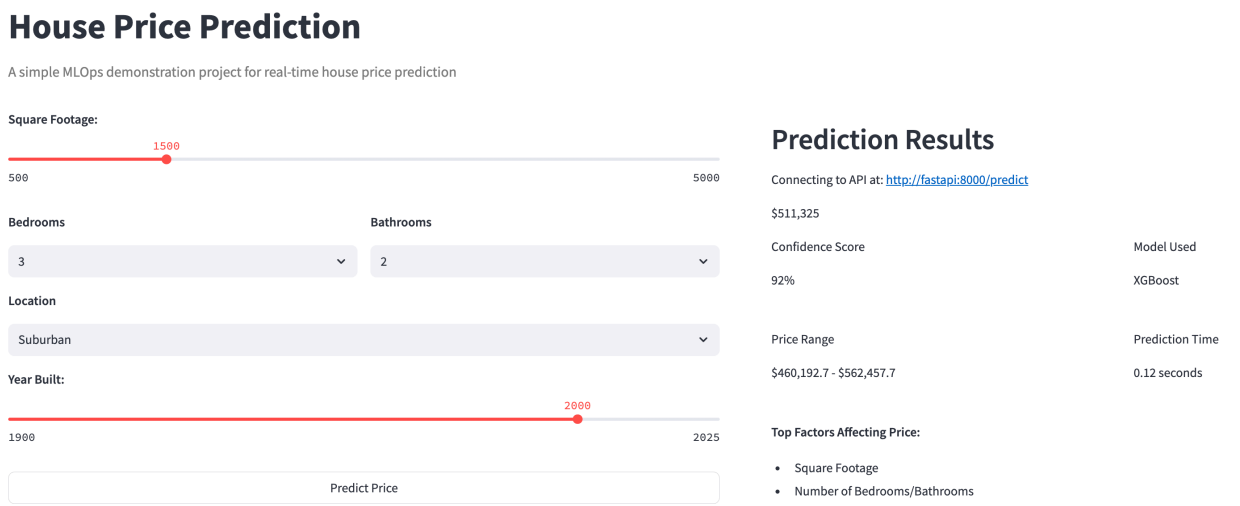
```
docker compose ps
```

[sample output]

```
CONTAINER ID  IMAGE                                 COMMAND               CREATED
STATUS      PORTS                     NAMES
16848c53ba20  localhost/initcron/fastapi:dev    uvicorn main:app ...  45
seconds ago  Created     0.0.0.0:8000->8000/tcp  house-price-
predictor_fastapi_1
2633c96a1682  localhost/initcron/streamlit:dev  streamlit run app...  45
seconds ago  Created     0.0.0.0:8501->8501/tcp  house-price-
predictor_streamlit_1
```

And you should be able to access the model and streamlit app respectively using

- http://localhost:8000/docs
- http://localhost:8501/



You should be able to make predictions using streamlit app connected to the model being serverd via fastapi.

Once you are done testing, clean it up using

```
docker compose down
```

Commit and Push the Changes

[check all the changes]

```
git status
```

[Sample output]

```
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    Dockerfile
    configs/
    data/processed/cleaned_house_data.csv
    data/processed/featured_house_data.csv
    docker-compose.yaml
    streamlit_app/Dockerfile
```

[commit and push to GitHub]

```
git add *
git commit -am "added Dockerfiles along with compose"
git push origin main
```

#courses/mlops