FastAPI For AI

🧠 1. Basic FastAPI Setup

```python
from fastapi import FastAPI

app = FastAPI()

@app.get("/")
def home():
    return {"msg": "FastAPI ML API is running!"}
```

📊 2. Regression & Classification API

```python
from pydantic import BaseModel
import joblib

# Load model
model = joblib.load("models/regression_model.pkl")

class InputFeatures(BaseModel):
    feature1: float
    feature2: float

@app.post("/predict_regression")
def predict(data: InputFeatures):
    x = [[data.feature1, data.feature2]]
    prediction = model.predict(x)
    return {"prediction": prediction.tolist()}
```

📄 3. NLP (Text Classification) API

```python
class TextIn(BaseModel):
    text: str

nlp_model = joblib.load("models/nlp_model.pkl")
vectorizer = joblib.load("models/vectorizer.pkl")

@app.post("/predict_nlp")
def classify_text(data: TextIn):
    vec = vectorizer.transform([data.text])
    result = nlp_model.predict(vec)
    return {"class": result[0]}
```

🖼 4. Computer Vision (Image Classification) API

```python
from fastapi import File, UploadFile
from PIL import Image
import io
import torchvision.transforms as transforms
import torch

cv_model = torch.load("models/image_model.pt")
cv_model.eval()

@app.post("/predict_image")
async def predict_image(file: UploadFile = File(...)):
    image_bytes = await file.read()
    image = Image.open(io.BytesIO(image_bytes))

    transform = transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.ToTensor(),
    ])
    img_tensor = transform(image).unsqueeze(0)

    with torch.no_grad():
        output = cv_model(img_tensor)
        pred = torch.argmax(output, dim=1).item()

    return {"prediction": pred}
```

🖹 5. Upload & Process CSV Files

```python
import pandas as pd
from fastapi import UploadFile, File

@app.post("/upload_csv")
async def upload_csv(file: UploadFile = File(...)):
    df = pd.read_csv(file.file)
    summary = df.describe().to_dict()
    return {"summary": summary}
```

---

6. Firebase Integration (Realtime DB or Firestore)

```python
import firebase_admin
from firebase_admin import credentials, firestore

cred = credentials.Certificate("firebase_key.json")
firebase_admin.initialize_app(cred)
db = firestore.client()

@app.post("/save_prediction")
def save(data: dict):
    doc_ref = db.collection("predictions").add(data)
    return {"status": "saved", "doc_id": doc_ref[1].id}
```

---

7. CORS Setup (Frontend Access)

```python
from fastapi.middleware.cors import CORSMiddleware

app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],  # Use specific domains in production
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)
```

---

8. Dockerfile for FastAPI

```
FROM python:3.10

WORKDIR /app

COPY . /app

RUN pip install --no-cache-dir -r requirements.txt

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

🔧 9. requirements.txt (Minimal)

```
fastapi
uvicorn
joblib
scikit-learn
pandas
torch
torchvision
Pillow
firebase-admin
python-multipart
```

🚀 10. Run FastAPI

Locally:

```
uvicorn main:app --reload
```

With Docker:

```
docker build -t my_fastapi_app .
docker run -p 8000:8000 my_fastapi_app
```