Flutter

Monday, October 7, 2024 7:50 PM

Main code Of binary text based model loading

```
from fastapi import FastAPI
from fastapi.middleware.cors import CORSMiddleware
from pydantic import BaseModel
import tensorflow as tf
import numpy as np
# Initialize FastAPI app
app = FastAPI()
# Enable CORS to allow requests from Flutter
app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],  # Allow all origins for testing, restrict in production
    allow_credentials=True,
    allow_methods=["*"],  # Allow all HTTP methods
    allow_headers=["*"],  # Allow all headers
)
# Load the model
try:
    model = tf.keras.models.load_model("heart_disease_model.h5")
    print("Model loaded successfully!")
except Exception as e:
    print(f"Error loading model: {e}")
# Input data model for request
class InputData(BaseModel):
    age: int
    sex: int
    cp: int
    trestbps: int
    chol: int
    fbs: int
    restecg: int
    thalach: int
    exang: int
    oldpeak: float
    slope: int
    ca: int
    thal: int
# Define a route to check the server status
@app.get("/")
def read_root():
    return {"message": "Welcome to the Heart Disease Predictor!"}
# Prediction route
@app.post("/predict")
def predict(data: InputData):
    try:
        # Prepare input data for the model
        input_data = np.array([[
            data.age, data.sex, data.cp, data.trestbps, data.chol, data.fbs,
            data.restecg, data.thalach, data.exang, data.oldpeak, data.slope,
            data.ca, data.thal
        ]])
        # Make the prediction
        prediction = model.predict(input_data)
        result = "Heart Disease Detected" if prediction[0][0] > 0.5 else "No
Heart Disease Detected"

        return {"prediction": result}
    except Exception as e:
        return {"error": str(e)}
```

If String type value available handle like that

```
from fastapi import FastAPI
from pydantic import BaseModel
from fastapi.middleware.cors import CORSMiddleware
import tensorflow as tf
import numpy as np

# Initialize FastAPI app
app = FastAPI()

# Enable CORS to allow requests from Flutter
app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],  # Allow all origins for testing, restrict in production
    allow_credentials=True,
    allow_methods=["*"],  # Allow all HTTP methods
    allow_headers=["*"],  # Allow all headers
)


# Load your model (change the path as needed)
try:
  model = tf.keras.models.load_model("your_model.h5")  # Replace with your model file
  print("Model loaded successfully!")
except Exception as e:
  print(f"Error loading model: {e}")

# Input data model for request
class InputData(BaseModel):
  age: int
  bp: int  # Blood pressure
  sg: float  # Specific gravity
  al: int  # Albumin
  su: int  # Sugar
  rbc: str  # Red blood cells (categorical)
  pc: str  # Pus cell (categorical)
  pcc: str  # Pus cell clumps (categorical)
  ba: str  # Bacteria (categorical)
  bgr: int  # Blood glucose random
  bu: float  # Blood urea
  sc: float  # Serum creatinine
  sod: float  # Sodium
  pot: float  # Potassium
  hemo: float  # Hemoglobin
  pcv: float  # Packed cell volume
  wc: float  # White blood cell count
  rc: float  # Red cell count
  htn: str  # Hypertension (categorical)
  dm: str  # Diabetes mellitus (categorical)
  cad: str  # Coronary artery disease (categorical)
  appet: str  # Appetite (categorical)
  pe: str  # Pedal edema (categorical)
  ane: str  # Anemia (categorical)

# Define a route to check the server status
@app.get("/")
def read_root():
  return {"message": "Welcome to the CKD Predictor!"}

# Prediction route
@app.post("/predict")
def predict(data: InputData):
```

```
    try:
        # Prepare input data for the model
        # Convert categorical features to numeric (e.g., one-hot encoding or label encoding)
        # Example encoding (replace these with the actual encoding used in your model)
        rbc_encoded = 1 if data.rbc == "present" else 0
        pc_encoded = 1 if data.pc == "present" else 0
        pcc_encoded = 1 if data.pcc == "present" else 0
        ba_encoded = 1 if data.ba == "present" else 0
        htn_encoded = 1 if data.htn == "yes" else 0
        dm_encoded = 1 if data.dm == "yes" else 0
        cad_encoded = 1 if data.cad == "yes" else 0
        appet_encoded = 1 if data.appet == "good" else 0
        pe_encoded = 1 if data.pe == "yes" else 0
        ane_encoded = 1 if data.ane == "yes" else 0

        input_data = np.array([[
            data.age,
            data.bp,
            data.sg,
            data.al,
            data.su,
            rbc_encoded,
            pc_encoded,
            pcc_encoded,
            ba_encoded,
            data.bgr,
            data.bu,
            data.sc,
            data.sod,
            data.pot,
            data.hemo,
            data.pcv,
            data.wc,
            data.rc,
            htn_encoded,
            dm_encoded,
            cad_encoded,
            appet_encoded,
            pe_encoded,
            ane_encoded,
        ]])

        # Make the prediction
        prediction = model.predict(input_data)

        # Customize this logic based on your model's output
        result = "CKD Detected" if prediction[0][0] > 0.5 else "No CKD Detected"

        return {"prediction": result}
    except Exception as e:
        return {"error": str(e)}
```

Environment Setup

```
# Create a virtual environment
python -m venv heart-disease-env

 # Activate the virtual environment

# Windows
heart-disease-env\Scripts\activate
```

```
# macOS/Linux
sourceheart-disease-env/bin/activate

# Install required packages
pip install fastapi uvicorn tensorflow numpy
```