go build

go: go.mod file not found in current directory or any parent directory; see 'go help modules'

go mod init eg/abhi

go: creating new go.mod: module eg/abhi

go: to add module requirements and sums:

go mod tidy

D:\openvino5>go build

Go script

package main

import (

    "bufio"

    "bytes"

    "encoding/json"

    "fmt"

    "io/ioutil"

    "net/http"

    "os"

)

type PredictionRequest struct {

    Smiles string `json:"smiles"`

}

func makePredictionRequest(apiURL, smiles string) {

    requestPayload := PredictionRequest{Smiles: smiles}

    jsonData, err := json.Marshal(requestPayload)

    if err != nil {

        fmt.Println("Error marshalling input data:", err)

        return

    }

    resp, err := http.Post(apiURL, "application/json", bytes.NewBuffer(jsonData))

    if err != nil {

        fmt.Println("Error making request:", err)

        return

    }

    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)

    if err != nil {

        fmt.Println("Error reading response body:", err)

        return

    }

    fmt.Println("Response from Flask API:", string(body))

}

func main() {

    flaskAPIURL := "http://localhost:5000/predict"

    scanner := bufio.NewScanner(os.Stdin)

    fmt.Println("Enter a SMILES string for lipophilicity prediction:")

    scanner.Scan()

    smilesString := scanner.Text()

    fmt.Println("Sending prediction request to Flask API...")

    makePredictionRequest(flaskAPIURL, smilesString)

}

Flask Python

from flask import Flask, request, jsonify

from rdkit import Chem

from rdkit.Chem import AllChem

import numpy as np

import openvino.runtime as ov

app = Flask(\_\_name\_\_)

# Load OpenVINO model

model\_path = 'lipophilicity\_openvino.xml'  # Replace with your model path

core = ov.Core()

compiled\_model = core.compile\_model(model\_path, "CPU")

# Function to convert SMILES to fingerprints

def smiles\_to\_fp(smiles, n\_bits=2048):

    mol = Chem.MolFromSmiles(smiles)

    fp = AllChem.GetMorganFingerprintAsBitVect(mol, radius=2, nBits=n\_bits)

    return np.array(fp)

# Endpoint for predictions

@app.route('/predict', methods=['POST'])

def predict():

    data = request.json

    smiles = data['smiles']

    fp = smiles\_to\_fp(smiles)

    input\_tensor = np.array([fp], dtype=np.float32)

    # OpenVINO inference

    ov\_input\_tensor = ov.Tensor(input\_tensor)

    result = compiled\_model([ov\_input\_tensor])[0]

    prediction = result[0]

    return jsonify({'prediction': prediction.tolist()})

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True, port=5000)