from django.shortcuts import render

from django.http import HttpResponse, HttpResponseRedirect

from django.urls import reverse

from django.utils import timezone

#from django.template import loader

from django.http import Http404, JsonResponse, HttpResponseServerError

from django.shortcuts import get\_object\_or\_404, render

# Create your views here.

from .models import Inventory

from .forms import PreferenceForm

from home.models import Item

###############################################################

from django.views.decorators.csrf import csrf\_exempt

from keras import models

import numpy as np

from keras.utils import to\_categorical

import os

import random

import json

###############################################################

def inventoryMain(request):

inventory = Inventory.objects.all()

context = {"inventory":inventory}

return render(request,"mainInventory.html",context)

def preferencesForm(request):

form = PreferenceForm()

context = {"form":form}

return render(request,"detail.html",context)

def prefOut(request):

if request.method =='POST':

form = PreferenceForm(request.POST)

if form.is\_valid():

price = form.cleaned\_data['highPrice']

cal = form.cleaned\_data['highCal']

ing = form.cleaned\_data['ingredient']

##################LOAD MODEL AND WEIGHTS#####################

#load json and create model

json\_file\_dir = os.path.dirname(\_\_file\_\_) # get current directory

file\_path = os.path.join(json\_file\_dir, 'model.json')

json\_file = open(file\_path,'r')

loaded\_model\_json = json\_file.read()

json\_file.close()

loaded\_model = models.model\_from\_json(loaded\_model\_json)

#load weights into new model

weights\_path = os.path.join(json\_file\_dir, 'weights.h5')

loaded\_model.load\_weights(weights\_path)

####################EVALUATE MODEL###########################

#Evaluate model on test data

loaded\_model.compile(loss='categorical\_crossentropy', # Cross-entropy

optimizer='rmsprop', # Root Mean Square Propagation

metrics=['accuracy']) # Accuracy performance metric

x = []

y = []

y\_items = []

for i in range(-1,2):

for j in range(-50,100,50):

x.append([(float(price)+(i\*1.0))/8.00,(int(cal)+ j)/1500,int(ing)])

x = np.array(x)

output = loaded\_model.predict(x)

for out in output:

max\_output = -1.00

output\_itemID = -1

for i in range(len(out)):

if out[i] > max\_output:

max\_output = out[i]

output\_itemID = i

output\_itemID+=1

y.append(output\_itemID)

y = set(y) #turns into a set of only unique elements

y = list(y) #turn back into a list

for id\_of\_item in y:

y\_items.append(Item.objects.get(pk=id\_of\_item))

context = {"output":y\_items}

return render(request,"netOutput.html",context)

def testComms(request):

if request.method == "GET":

num = random.randint(100,500)

num = str(num)

char = random.choice(["A","B","C","D","E","F"])

code = char + num

return HttpResponse(code,content\_type="text/plain")

def testJSON(request):

if request.method == "GET":

json\_dict = {}

for item in list(Item.objects.all()):

json\_dict.update({(f'Item {item.id}'):{

"Item Name":item.item\_name,

"Price":item.price,

"Ingredients":item.ingredients}})

return JsonResponse(json\_dict)

@csrf\_exempt

def testPOST(request):

#if request.is\_ajax():

#if request.method == "POST":

#print(f'Raw Data: {request.body}')

if request.method == "POST":

#json\_data = json.loads(request.body) # request.raw\_post\_data w/ Django < 1.4

try:

#data = json\_data['data']

print(json.loads(request.body.decode('UTF-8')))

except KeyError:

return HttpResponseServerError("Malformed data!")

return JsonResponse({"Hey Cutie!":"I got your data"})