

```
"""Untitled0.ipynb
```

```
Automatically generated by Colaboratory.
```

```
Original file is located at
```

```
https://colab.research.google.com/drive/1PYFZ7zKhWpFF5YilnguhZ8X1EgtSIJN4
```

```
"""
```

```
import re
```

```
import numpy as np
```

```
import os
```

```
from flask import Flask, app,request,render_template
```

```
import sys
```

```
from flask import Flask, request, render_template, redirect, url_for
```

```
import argparse
```

```
from tensorflow import keras
```

```
from PIL import Image
```

```
from timeit import default_timer as timer
```

```
import test
```

```
from pyngrok import ngrok
```

```
import pandas as pd
```

```
import numpy as np
```

```
import random
```

```
def get_parent_dir(n=1):
```

```
    """ returns the n-th parent directory of the current
```

```
    working directory """
```

```
    current_path = os.path.dirname(os.path.abspath(__file__))
```

```
    for k in range(n):
```

```
        current_path = os.path.dirname(current_path)
```

```
    return current_path
```

```
src_path=r'/content/drive/MyDrive/IBM_PROJECT/yolo_structure/2_Training/src'
print(src_path)

utils_path=r'/content/drive/MyDrive/IBM_PROJECT/yolo_structure/Utils'
print(utils_path)
```

```
sys.path.append(src_path)
sys.path.append(utils_path)
```

```
import argparse

from keras_yolo3.yolo import YOLO, detect_video

from PIL import Image

from timeit import default_timer as timer

from utils import load_extractor_model, load_features, parse_input, detect_object

import test

import utils

import pandas as pd

import numpy as np

from Get_File_Paths import GetFileList

import random
```

```
os.environ["TF_CPP_MIN_LOG_LEVEL"] = "3"
```

```
# Set up folder names for default values

data_folder = os.path.join(get_parent_dir(n=1), "yolo_structure", "Data")
```

```
image_folder = os.path.join(data_folder, "Source_Images")
```

```
image_test_folder = os.path.join(image_folder, "Test_Images")
```

```
detection_results_folder = os.path.join(image_folder, "Test_Image_Detection_Results")
detection_results_file = os.path.join(detection_results_folder, "Detection_Results.csv")
```

```
model_folder = os.path.join(data_folder, "Model_Weights")
```

```
model_weights = os.path.join(model_folder, "trained_weights_final.h5")
```

```
model_classes = os.path.join(model_folder, "data_classes.txt")
```

```
anchors_path = os.path.join(src_path, "keras_yolo3", "model_data", "yolo_anchors.txt")
```

```
FLAGS = None
```

```
from cloudant.client import Cloudant
```

```
# Authenticate using an IAM API key
```

```
client =
```

```
Cloudant.iam('ef7f4729-2486-45c5-a7fa-f4140373e2e6-bluemix','6GfFjs3engXLnSJB8Kp4fbs7H  
TKwrJpWJE7wNPGzZPVW', connect=True)
```

```
# Create a database using an initialized client
```

```
my_database = client.create_database('my_database')
```

```
app=Flask(__name__)
```

```
port_no=5000
```

```
ngrok.set_auth_token("2H7aM94zEuTa40t3J6jKpIqWAc3_B2UxzZs6qxehtgadxQW")
```

```
public_url = ngrok.connect(port_no).public_url
```

```
print(f"To acces the Gloable link please click {public_url}")
```

```
#default home page or route
```

```
@app.route('/')  
  
def index():  
  
    return render_template('index.html')
```

```
@app.route('/index.html')  
  
def home():  
  
    return render_template("index.html")
```

#registration page

```
@app.route('/register')  
  
def register():  
  
    return render_template('register.html')
```

```
@app.route('/afterreg', methods=['POST'])  
  
def afterreg():  
  
    x = [x for x in request.form.values()]  
  
    print(x)  
  
    data = {  
  
        '_id': x[1], # Setting _id is optional  
  
        'name': x[0],  
  
        'psw':x[2]  
  
    }  
  
    print(data)  
  
  
    query = {'_id': {'$eq': data['_id']}}  
  
  
  
  
    docs = my_database.get_query_result(query)  
  
    print(docs)
```

```

print(len(docs.all()))

if(len(docs.all())==0):

    url = my_database.create_document(data)

    #response = requests.get(url)

    return render_template('register.html', pred="Registration Successful, please login using
your details")
else:

    return render_template('register.html', pred="You are already a member, please login using
your details")


#login page

@app.route('/login')

def login():

    return render_template('login.html')


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

def afterlogin():

    user = request.form['_id']

    passw = request.form['psw']

    print(user,passw)

    query = {'_id': {'$eq': user}}

    docs = my_database.get_query_result(query)

    print(docs)

    print(len(docs.all()))

    if(len(docs.all())==0):

        return render_template('login.html', pred="The username is not found.")
    else:

        if((user==docs[0][0]['_id'] and passw==docs[0][0]['psw'])):

```

```
        return redirect(url_for('prediction'))

    else:

        print('Invalid User')
```

```
@app.route('/logout')

def logout():

    return render_template('logout.html')
```

```
@app.route('/prediction')

def prediction():

    return render_template('prediction.html',path="../static/img/6623.jpg",)
```

```
@app.route('/result',methods=["GET","POST"])

def res():

    # Delete all default flags

    parser = argparse.ArgumentParser(argument_default=argparse.SUPPRESS)

    """

    Command line options

    """

    f = request.files['file']

    f.save("./drive/MyDrive/IBM_PROJECT/Flask/static/img/"+f.filename)
```

```
parser.add_argument(

    "--input_path",

    type=str,

    default=image_test_folder,

    help="Path to image/video directory. All subdirectories will be included. Default is "
```

```
+ image_test_folder,  
)
```

```
parser.add_argument(  
    "--output",  
    type=str,  
    default=detection_results_folder,  
    help="Output path for detection results. Default is "  
    + detection_results_folder,  
)
```

```
parser.add_argument(  
    "--no_save_img",  
    default=False,  
    action="store_true",  
    help="Only save bounding box coordinates but do not save output images with annotated  
boxes. Default is False.",  
)
```

```
parser.add_argument(  
    "--file_types",  
    "--names-list",  
    nargs="*",  
    default=[],  
    help="Specify list of file types to include. Default is --file_types .jpg .jpeg .png .mp4",  
)
```

```
parser.add_argument(  
    "--yolo_model",  
    type=str,  
    dest="model_path",  
    default=model_weights,  
    help="Path to pre-trained weight files. Default is " + model_weights,
```

)

```
parser.add_argument(  
    "--anchors",  
    type=str,  
    dest="anchors_path",  
    default=anchors_path,  
    help="Path to YOLO anchors. Default is " + anchors_path,  
)
```

```
parser.add_argument(  
    "--classes",  
    type=str,  
    dest="classes_path",  
    default=model_classes,  
    help="Path to YOLO class specifications. Default is " + model_classes,  
)
```

```
parser.add_argument(  
    "--gpu_num", type=int, default=1, help="Number of GPU to use. Default is 1"  
)
```

```
parser.add_argument(  
    "--confidence",  
    type=float,  
    dest="score",  
    default=0.25,  
    help="Threshold for YOLO object confidence score to show predictions. Default is 0.25.",  
)
```

```
parser.add_argument(  
    "--box_file",
```



```

type=str,
dest="box",
default=detection_results_file,
help="File to save bounding box results to. Default is "
+ detection_results_file,
)

```

```

parser.add_argument(
    "--postfix",
    type=str,
    dest="postfix",
    default="_disease",
    help='Specify the postfix for images with bounding boxes. Default is "_disease"',
)

```

```

yolo = YOLO(
    **{
        "model_path": FLAGS.model_path,
        "anchors_path": FLAGS.anchors_path,
        "classes_path": FLAGS.classes_path,
        "score": FLAGS.score,
        "gpu_num": FLAGS.gpu_num,
        "model_image_size": (416, 416),
    }
)

```

```

img_path="/drive/MyDrive/IBM_PROJECT/Flask/static/img/"+f.filename

```

```
prediction, image,lat,lon= detect_object(

    yolo,

    img_path,

    save_img=save_img,

    save_img_path=FLAGS.output,

    postfix=FLAGS.postfix,

)

yolo.close_session()

return

render_template('prediction.html',prediction=str(prediction),path="../static/img/"+f.filename)


""" Running our application """

if __name__ == "__main__":

    app.run(port=port_no)
```