STRAWBERRY SEGMENTATION

Nada Belaidi

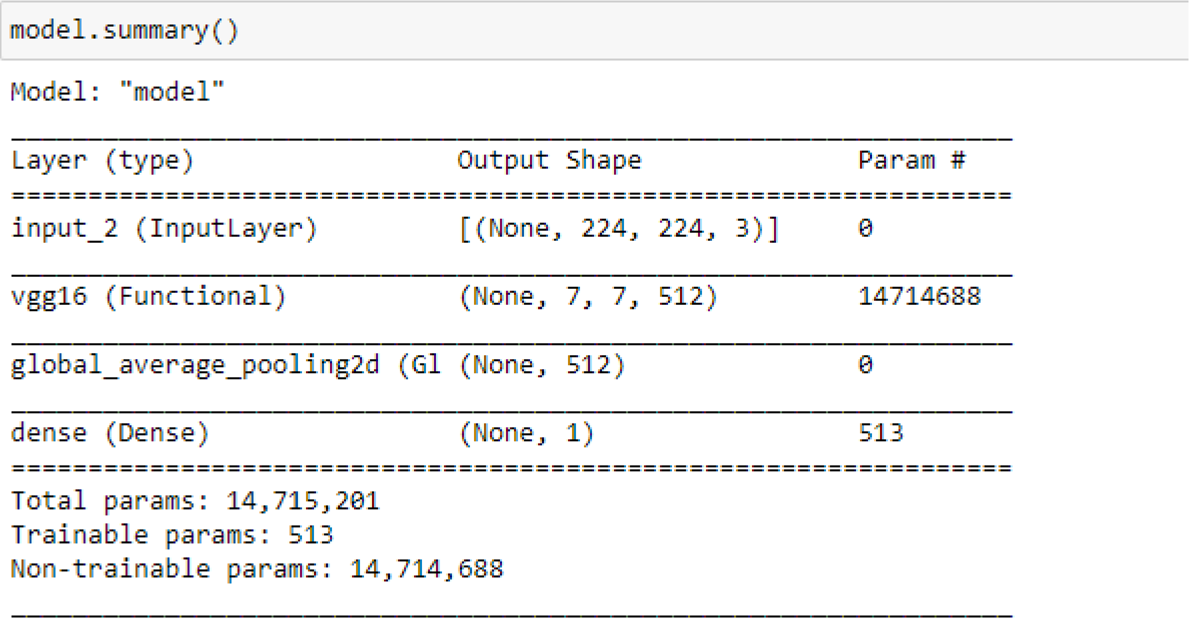
Data Glacier 04/072021

For week4's assignment as a Data Glacier intern, I chose to make a simple strawberry segmentation app. I'm using Flask and a pretrained model I already have.

STEPS

CHOOSING THE MODEL :

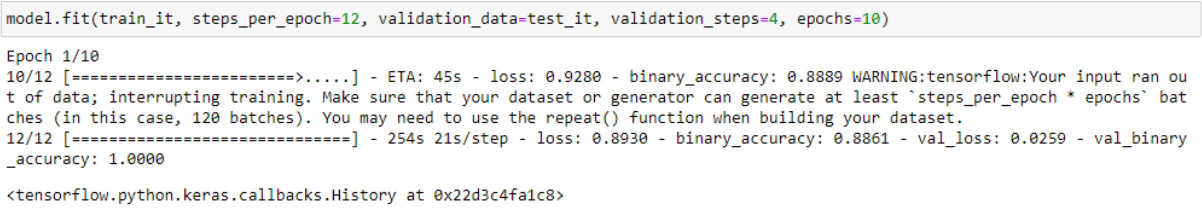
This an ImageNet pre-trained model that I'm going to use to segment strawberries out of other random items.



```
model.summary()
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
vgg16 (Functional)           (None, 7, 7, 512)         14714688
_________________________________________________________________
global_average_pooling2d (Gl (None, 512)               0
_________________________________________________________________
dense (Dense)                (None, 1)                 513
=================================================================
Total params: 14,715,201
Trainable params: 513
Non-trainable params: 14,714,688
_________________________________________________________________
```

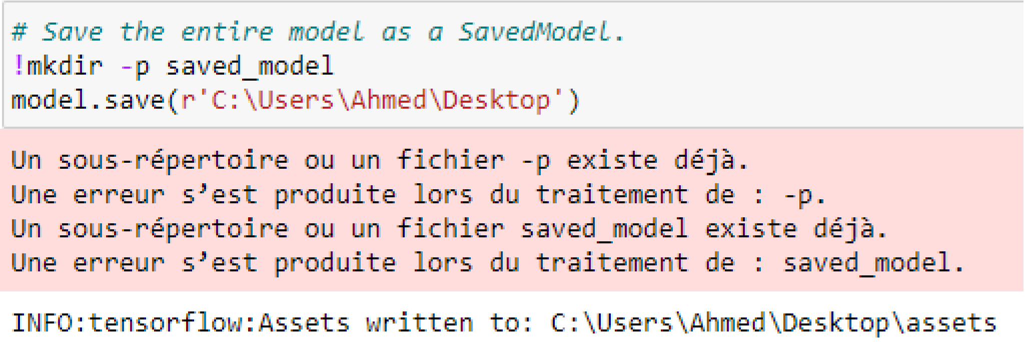
First, I trained the model:



```
model.fit(train_it, steps_per_epoch=12, validation_data=test_it, validation_steps=4, epochs=10)
```

```
Epoch 1/10
10/12 [========================>.....] - ETA: 45s - loss: 0.9280 - binary_accuracy: 0.8889 WARNING:tensorflow:Your input ran ou
t of data; interrupting training. Make sure that your dataset or generator can generate at least `steps_per_epoch * epochs` bat
ches (in this case, 120 batches). You may need to use the repeat() function when building your dataset.
12/12 [==============================] - 254s 21s/step - loss: 0.8930 - binary_accuracy: 0.8861 - val_loss: 0.0259 - val_binary
_accuracy: 1.0000

<tensorflow.python.keras.callbacks.History at 0x22d3c4fa1c8>
```

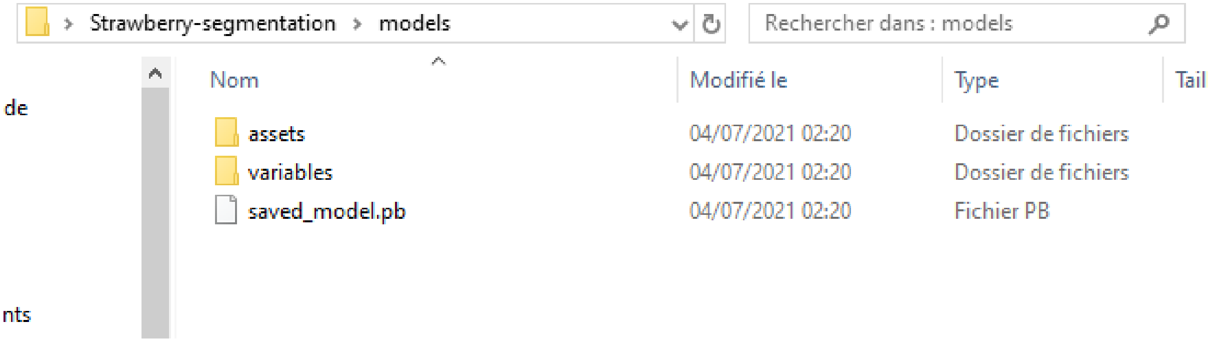
And then I saved it.



```
# Save the entire model as a SavedModel.
!mkdir -p saved_model
model.save(r'C:\Users\Ahmed\Desktop')
```

```
Un sous-répertoire ou un fichier -p existe déjà.
Une erreur s’est produite lors du traitement de : -p.
Un sous-répertoire ou un fichier saved_model existe déjà.
Une erreur s’est produite lors du traitement de : saved_model.
```

```
INFO:tensorflow:Assets written to: C:\Users\Ahmed\Desktop\assets
```



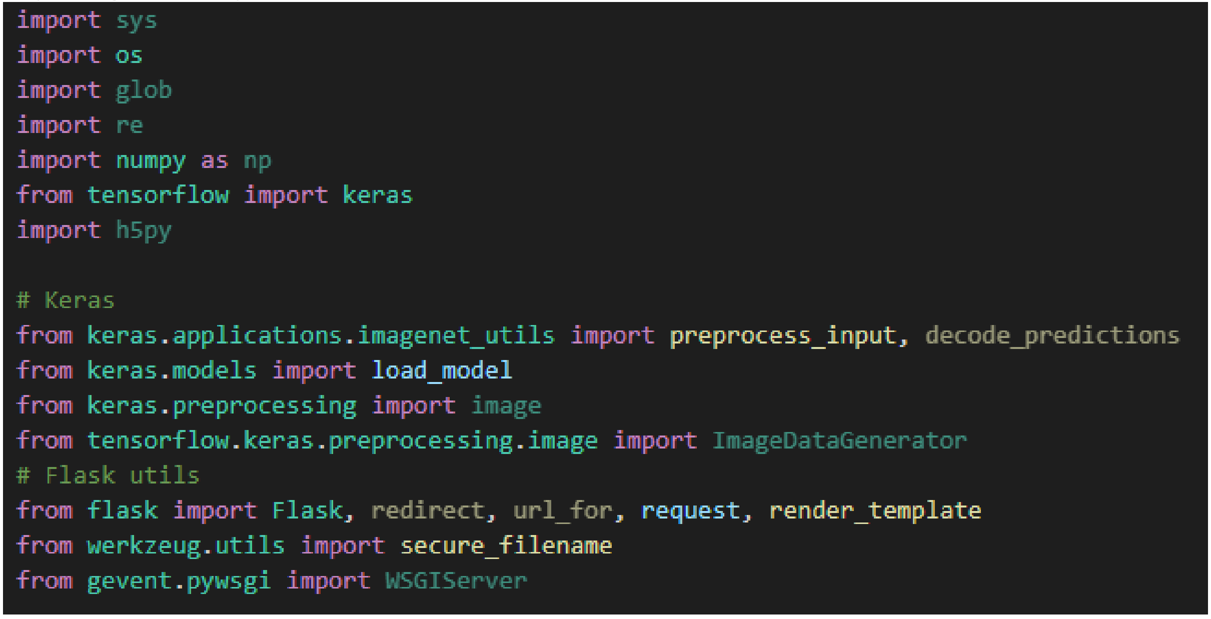
Nom	Modifié le	Type	Tail
assets	04/07/2021 02:20	Dossier de fichiers	
variables	04/07/2021 02:20	Dossier de fichiers	
saved_model.pb	04/07/2021 02:20	Fichier PB	

INSTALLING FLASK AND THE VIRTUAL ENVIRONEMENT:

py -m pip install venv venv
py -m pip install flask
venv\Scripts\activate to activate the environement.

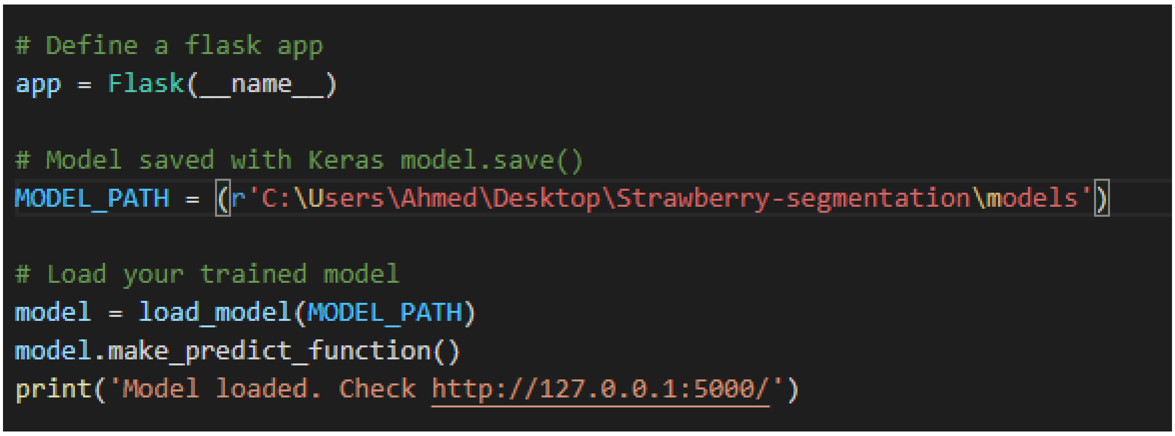
CREATING THE APP:

Importing necessary modules.



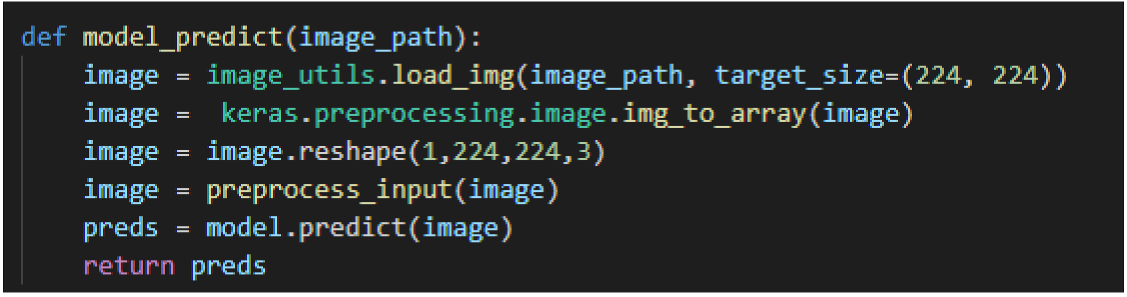
```
import sys
import os
import glob
import re
import numpy as np
from tensorflow import keras
import h5py

# Keras
from keras.applications.imagenet_utils import preprocess_input, decode_predictions
from keras.models import load_model
from keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator
# Flask utils
from flask import Flask, redirect, url_for, request, render_template
from werkzeug.utils import secure_filename
from gevent.pywsgi import WSGIServer
```

Defining the app and loading the model:



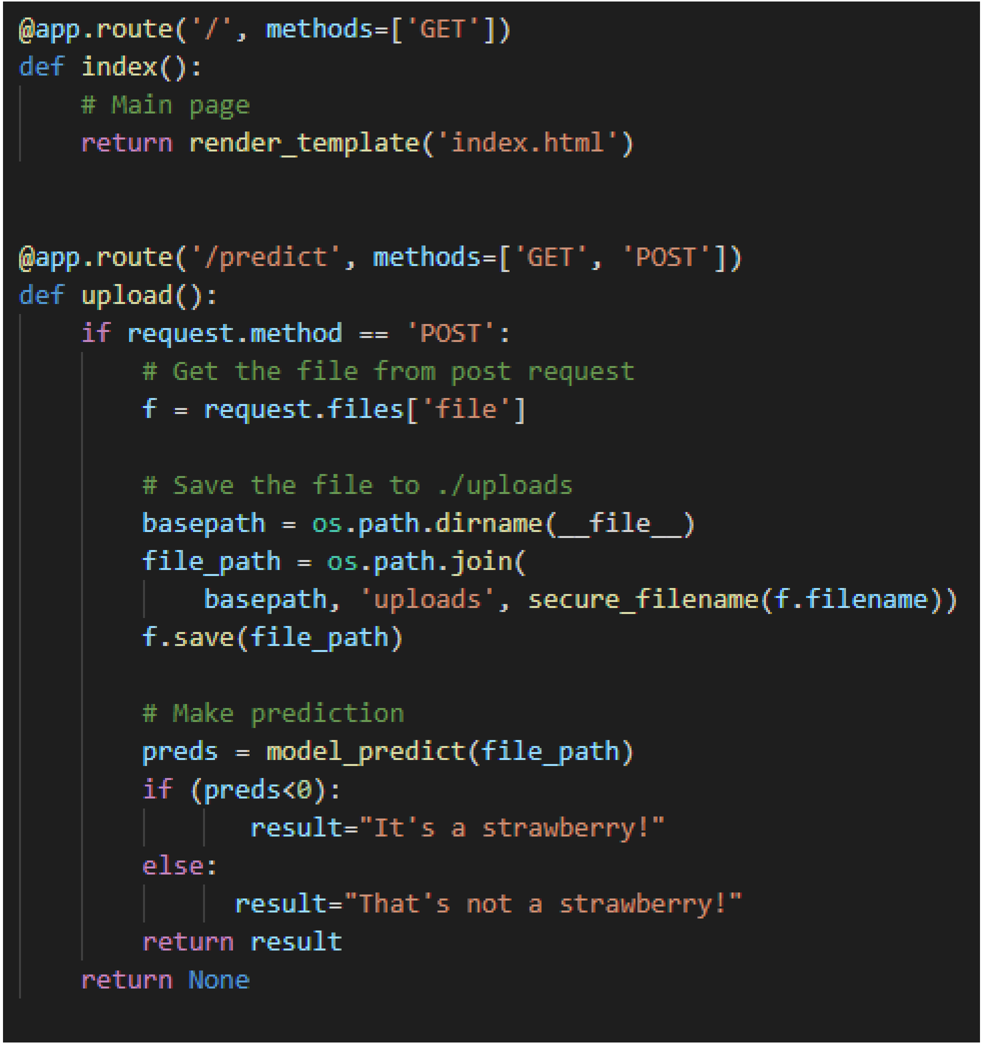
```python
# Define a flask app
app = Flask(__name__)

# Model saved with Keras model.save()
MODEL_PATH = (r'C:\Users\Ahmed\Desktop\Strawberry-segmentation\models')

# Load your trained model
model = load_model(MODEL_PATH)
model.make_predict_function()
print('Model loaded. Check http://127.0.0.1:5000/')
```

Defining the model_predict function:



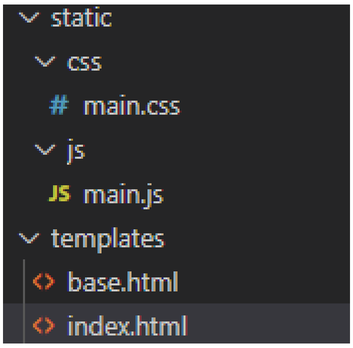
```python
def model_predict(image_path):
    image = image_utils.load_img(image_path, target_size=(224, 224))
    image =  keras.preprocessing.image.img_to_array(image)
    image = image.reshape(1,224,224,3)
    image = preprocess_input(image)
    preds = model.predict(image)
    return preds
```

Defining the methods:



```python
@app.route('/', methods=['GET'])
def index():
    # Main page
    return render_template('index.html')


@app.route('/predict', methods=['GET', 'POST'])
def upload():
    if request.method == 'POST':
        # Get the file from post request
        f = request.files['file']

        # Save the file to ./uploads
        basepath = os.path.dirname(__file__)
        file_path = os.path.join(
            basepath, 'uploads', secure_filename(f.filename))
        f.save(file_path)

        # Make prediction
        preds = model_predict(file_path)
        if (preds<0):
            result="It's a strawberry!"
        else:
            result="That's not a strawberry!"
        return result
    return None
```

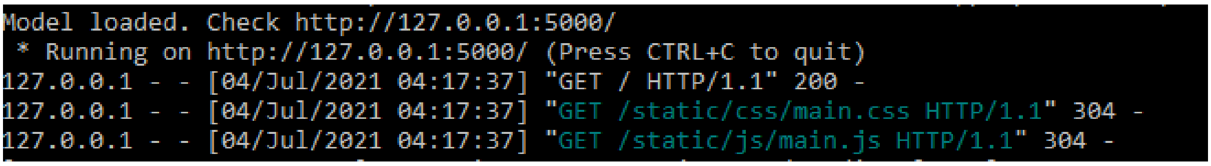
FRONT END DEVELOPEMENT :



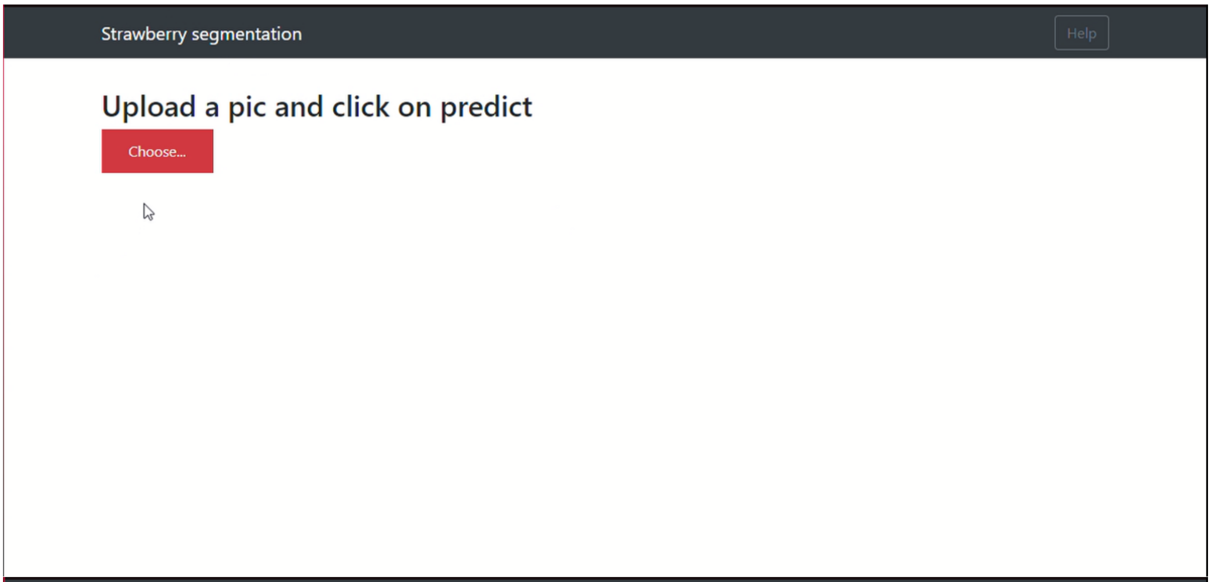
```
static
  css
    main.css
  js
    main.js
templates
  base.html
  index.html
```

RUNNING THE APP :

Set FLASK_APP=app.py
Set FLASK_DEBUG=1
Flask run

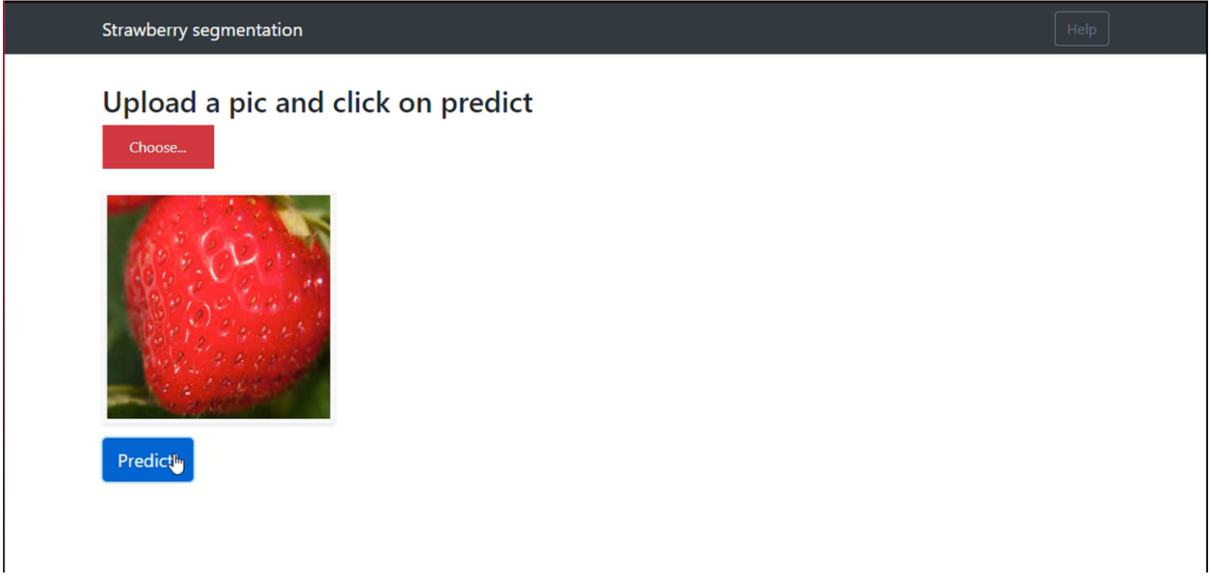


```
Model loaded. Check http://127.0.0.1:5000/
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [04/Jul/2021 04:17:37] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [04/Jul/2021 04:17:37] "GET /static/css/main.css HTTP/1.1" 304 -
127.0.0.1 - - [04/Jul/2021 04:17:37] "GET /static/js/main.js HTTP/1.1" 304 -
```



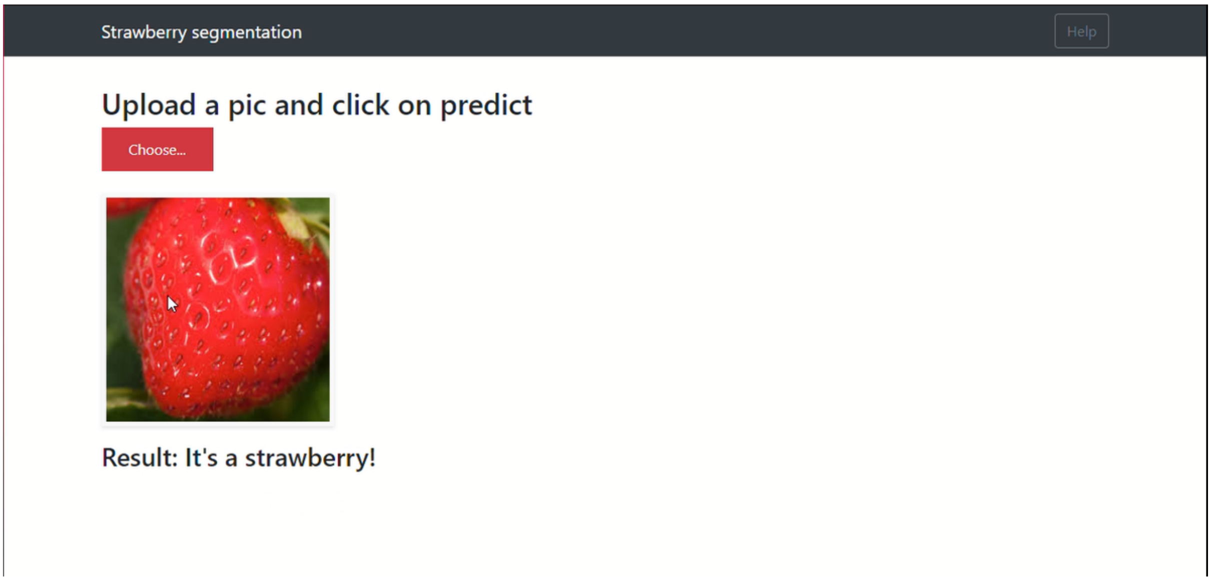
Strawberry segmentation

Upload a pic and click on predict



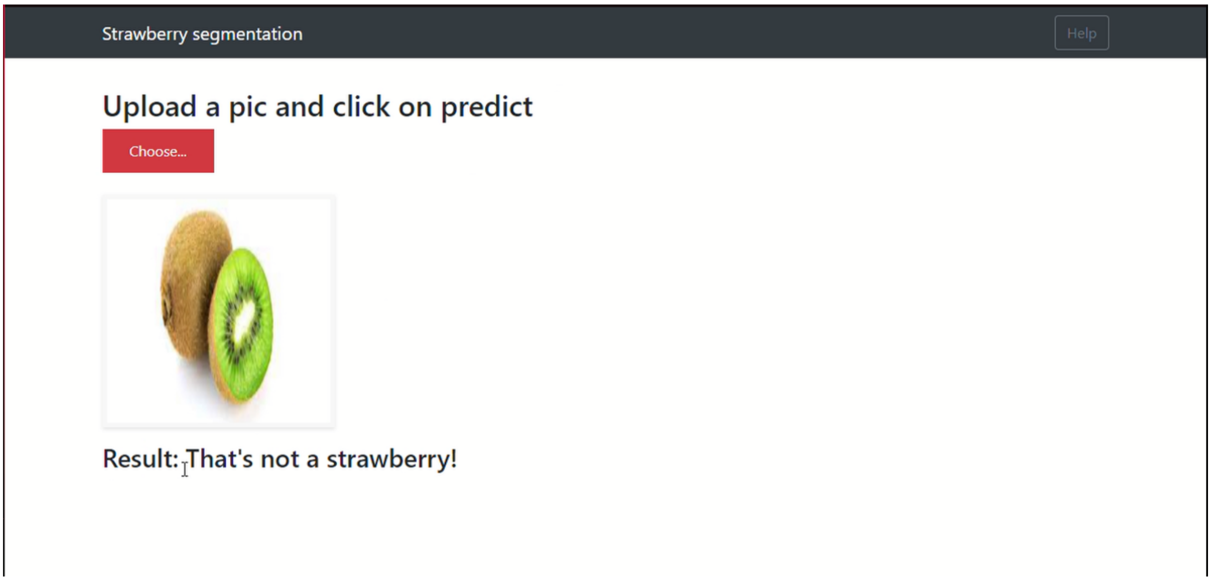
Strawberry segmentation

Upload a pic and click on predict



Upload a pic and click on predict

Result: It's a strawberry!



Upload a pic and click on predict

Result: That's not a strawberry!

Last step is pushing the code to Github.