**Template for setting up a new deployable project at GCP (Flask at GAE)**

notes:

* 'natality' is project name. replace it with a new name.
* editing html files is trivial.
* if editing html and main.py seems easy, do evth in a Vertex Notebook and use Cloud Shell only for debugging
* this template developed based on this series of posts:

<https://medium.com/@nutanbhogendrasharma/deploy-machine-learning-model-in-google-cloud-platform-using-flask-part-3-20db0037bdf8>

**General workflow:**

First, create a project folder like 'pg\_natality' in project\_repos folder. put there empty folder cloned frm GH. build a model there, name it like tit\_model.ipynb. Put model artifact into /<project>-app/ folder.

**To make deployment, follow this guide:**

mkdir natality-app

cd natality-app

pip install virtualenv

virtualenv natality-app-venv

source natality-app-venv/bin/activate

pip install Flask

nano main.py

#import Flask

from flask import Flask

#create an instance of Flask

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

@app.route('/')

def home():

return "Hello World"

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

app.run(debug=True)

python main.py

mkdir templates

cd templates

nano home.html

<!doctype html>

<html>

<head>

<title> Predict Baby Weight </title>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<link rel="stylesheet" href="<https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css>">

<script src="<https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js>"></script>

<script src="<https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js>"></script>

</head>

<body>

<div class="container">

<div class="row my-5 pl-3">

<h1>Predict Baby Weight</h1>

</div>

<!-- Starts form section -->

<div class="form-container ">

<form class="form-horizontal" action = "/predict/" method="post">

<div class="form-group row">

<label class="control-label col-sm-2" for="is\_male">Is male? (Enter 1 or 0):</label>

<div class="col-sm-4">

<input type="text" class="form-control" id="is\_male" name="is\_male">

</div>

</div>

<div class="form-group row">

<label class="control-label col-sm-2" for="mother\_age">Mother's age:</label>

<div class="col-sm-4">

<input type="text" class="form-control" id="mother\_age" name="mother\_age">

</div>

</div>

<div class="form-group row">

<label class="control-label col-sm-2" for="plurality">Plurality:</label>

<div class="col-sm-4">

<input type="text" class="form-control" id="plurality" name="plurality">

</div>

<div class="form-group row">

<label class="control-label col-sm-2" for="gestation\_weeks">Gestation (in weeks):</label>

<div class="col-sm-4">

<input type="text" class="form-control" id="gestation\_weeks" name="gestation\_weeks">

</div>

</div>

<div class="form-group row">

<label class="control-label col-sm-2" for="">&nbsp;</label>

<div class="col-sm-offset-2 col-sm-4">

<button type="submit" class="btn btn-primary">Predict</button>

</div>

</div>

</form>

<!-- Ends form section -->

</div>

</div>

</body>

</html>

cd ..

nano main.py

#import Flask

from flask import Flask, render\_template

#create an instance of Flask

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

@app.route('/')

def home():

return render\_template('home.html')

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

app.run(debug=True)

(to check http locally, can use lynx <http://127.0.0.1:5000> -dump as an alternative to baseline curl)

(notice that lynx pings dynamically, so you do not have to run it repeatedly)

cd templates

nano predict.html

<!doctype html>

<html>

<head>

<title> Prediction </title>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<link rel="stylesheet" href="<https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css>">

<script src="<https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js>"></script>

<script src="<https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js>"></script>

</head>

<body>

<div class="container">

<div class="row my-5 pl-3">

<h1>Prediction is {{ prediction }}</h1>

</div>

</div>

</body>

</html>

(put the model in the root project folder)

pip install numpy

pip install joblib

pip install sklearn==0.0

pip install xgboost

pip install pandas

pip install --upgrade google-cloud-storage

pip install yfinance

nano main.py

#import Flask

from flask import Flask, render\_template, request

#create an instance of Flask

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

@app.route('/')

def home():

return render\_template('home.html')

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

def predict():

if request.method == "POST":

#get form data

tv = request.form.get('tv')

radio = request.form.get('radio')

newspaper = request.form.get('newspaper')

#call preprocessDataAndPredict and pass inputs

try:

prediction = preprocessDataAndPredict(tv, radio, newspaper)

#pass prediction to template

return render\_template('predict.html', prediction = prediction)

except ValueError:

return "Please Enter valid values"

pass

pass

def preprocessDataAndPredict(tv, radio, newspaper):

test\_data = [tv, radio, newspaper]

print(test\_data)

test\_data = np.array(test\_data).astype(np.float)

test\_data = test\_data.reshape(1,-1)

print(test\_data)

file = open("lr\_model.pkl","rb")

trained\_model = joblib.load(file)

prediction = trained\_model.predict(test\_data)

return prediction

pass

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

app.run(debug=True)

(run locally, make sure it works)

[from project root] pip freeze > requirements.txt

(you need 10-15 packages in requirements. remove some if there are too many)

nano app.yaml

runtime: python38

service: "<project name>"

gcloud init

gcloud app deploy

Debugging notes:

if app works locally, but not on gae, make sure that requirements.txt contains all dependencies.

gae console > services has logs for each service. this decreases need to ever use cloud shell.

f1 instance of gae may have inusfficient ram for some use cases.

i can put my own files into gae app folder. this will work as long as their path is specified relative to that folder.