notebook

April 15, 2021

1 Web App Deployment Using Flask

Name: Omar Safwat

Batch Code: LISP01

Date: 2021-03-28

1.1 Introduction

This report presents my approach in deploying my web app using Flask, the micro Web app frame work on python. As a demonstration, the app asks the client to select a polynomial degree from a top down list, and subsequently plot a polynomial regression on the airquality data set.

1.2 Importing libraries

[ ]:
```
from flask import Flask, request, render_template #Web app framework
import numpy as np
import pandas as pd
#Scikit learn for regression model
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
import matplotlib.pyplot as plt #To create plots
from io import BytesIO #To encode plots and pass them to render_template
import base64
```

1.3 Home page

The home page prompts the client to choose an order for the degree polynomial to build the model. A HTTP GET request is issued (refer to layout.html), and the url is then parsed using the imported function *request()* from Flask, as seen below.

[ ]:
```
#Create app using flask
app = Flask(__name__)

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

```
def index():
    #Parse html for "GET" request with client's input
    poly_order = request.args.get("poly_order", 1, type=int)
    #Load data
    airquality = pd.read_csv('airquality.csv')
    airquality.dropna(inplace=True)
    data = airquality[['Temp', 'Ozone']]
    #Build  model
    plot_url, r2 = build_model(poly_order, data)
    return render_template('layout.html', tables=[data.head(6).
 ↪to_html(classes='data', header=True)], r2=r2, plot_url=plot_url.
 ↪decode('utf8'), chosen_order=poly_order)
```

1.4 Model building

The client's selection is then passed to the function *build_model()* where the polynomial regression model is built as seen below.

[ ]:
```
def build_model(poly_order, data):

    x = data['Ozone']
    x = x[:, np.newaxis]
    y = data['Temp']
    y = y[:, np.newaxis]
    #Create polynomial terms out of 1D array
    poly_features = PolynomialFeatures(degree = poly_order)
    x_poly = poly_features.fit_transform(x)
    #Build the model
    model = LinearRegression()
    model.fit(x_poly, y)
    y_poly_pred = model.predict(x_poly)

    #Training Error
    r2 = r2_score(y, y_poly_pred)

    #Sort axis and create plot
    plt.scatter(data['Ozone'], data['Temp'], s=10)
    sorted_zip = sorted(zip(data['Ozone'], y_poly_pred))
    x, y_poly_pred = zip(*sorted_zip)
    plt.plot(x, y_poly_pred, color='m')
    plt.xlabel('Ozone (ppb)')
    plt.ylabel('Temp (Fahrenheit)')
    #Save plot to return to html
    img = BytesIO()
    plt.savefig(img, format='png')
    plt.close()
    img.seek(0)
```

```
    #Encode figure and pass to html in index function
    plot_url = base64.b64encode(img.getvalue())
    return(plot_url, r2)

#Run app with debug
if __name__ == "__main__":
    app.run(debug=True)
```

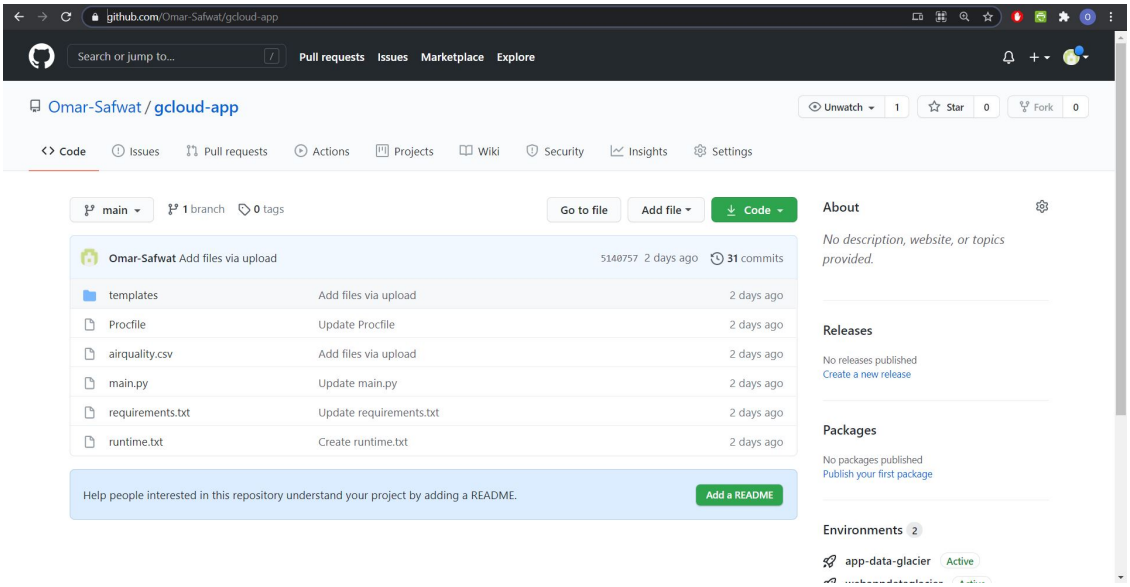
2 Deployment on Heroku

The App was then deployed on Heroku with the following steps: 1. Upload the following to version control: * **templates**: Folder containing HTML for app. * **main.py**: Flask web app python module.

2. Add the necessary files for deployment on Heroku:
   - **Procfile**: Instructs Heroku how to run the app.
   - **runtime.txt**: Specify version of python needed to run the app.
   - **requirements.txt**: Tells Heroku which packages to install for the app.
3. After creating a new app instance on Heroku, link Heroku to the repo.
4. Build and deploy the app.

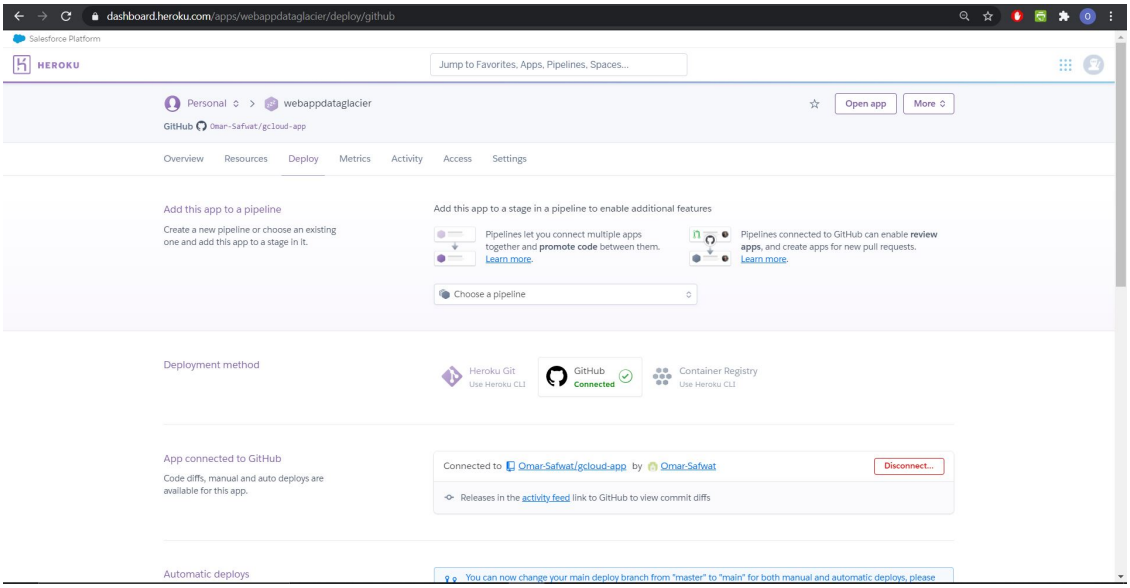
[2]:
```
from IPython.display import Image
Image(filename= "repoContent.jpg", width=900)
```

[2]:

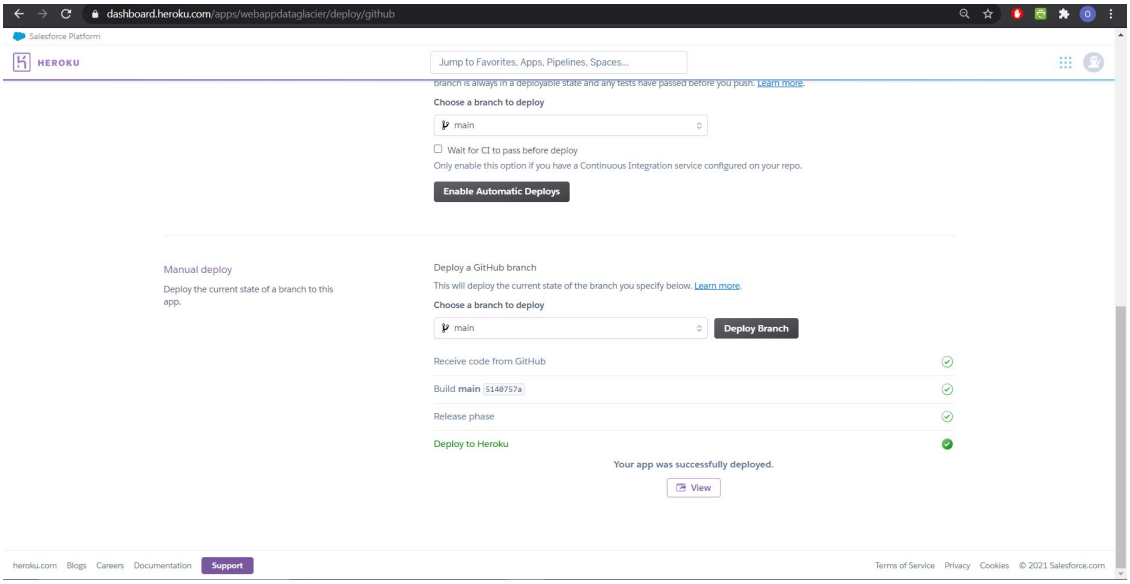


[4]:
```
Image(filename= "connectedToRepo.jpg", width=900)
```

[4]:

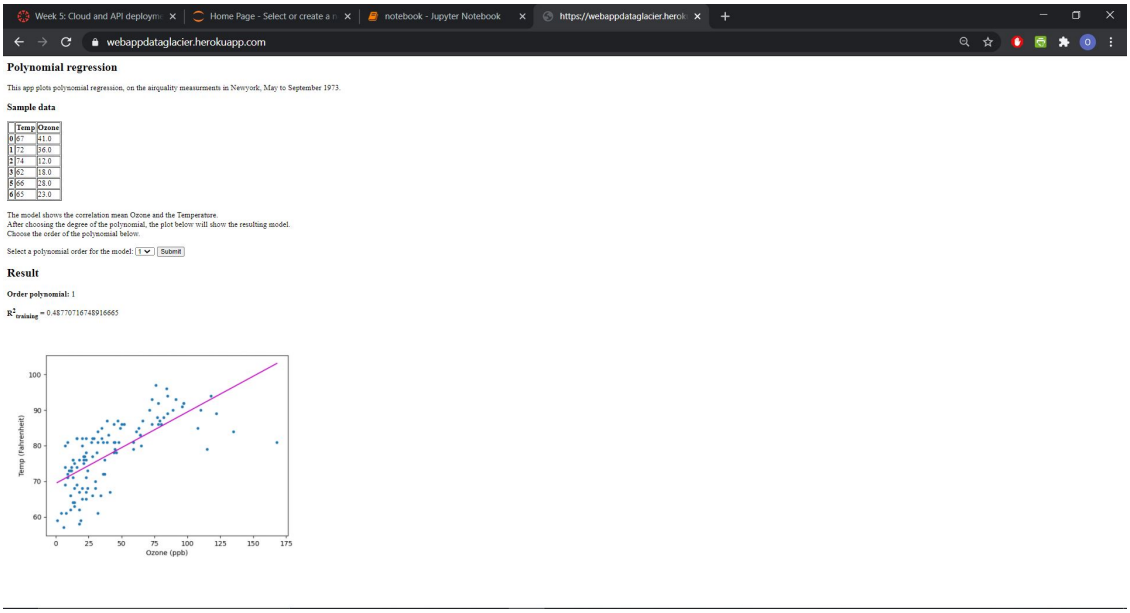


[5]:
```
Image(filename= "builtSuccessfully.jpg", width=900)
```

[5]:


[3]:
```
Image(filename= "API.jpg", width=900)
```

[3]:


Week 5: Cloud and API deploym
Home Page - Select or create a n
notebook - Jupyter Notebook
https://webappdataglacier.herok
webappdataglacier.herokuapp.com
Polynomial regression
This app plots polynomial regression, on the airquality measurments in Newyork, May to September 1973.
Sample data
Temp Ozone
0 67 41.0
1 72 36.0
2 74 12.0
3 62 18.0
4 66 28.0
6 65 23.0
The model shows the correlation mean Ozone and the Temperature.
After choosing the degree of the polynomial, the plot below will show the resulting model.
Choose the order of the polynomial below.
Select a polynomial order for the model: 1 Submit
Result
Order polynomial: 1
R2training = 0.48770716748916665
Temp (Fahrenheit)
Ozone (ppb)