

Deployment of a flask app using Heroku

Name: Jamila Hamdi

Batch code: LISUM01

Submission date: 11/07/2021

Submitted to: Data Glacier

Abstract

This project aims to deploy a Flask application to Heroku by hosting it on the internet in order to provide a public URL where anyone can view the work or product.

What is Heroku?

Heroku is a Platform as a service (PaaS), it lets us deploy, run and manage applications written in Ruby, Node.js, Java, Python, Clojure, Scala, Go and PHP.

In our case, the application is written in Python.

Steps of deployment:

- 1. Prerequisites:**
- 2. Delpoy the app on Heroku**
- 3. Testing the app**

Let's start!

1. Prerequisites:

- Heroku account (on <https://www.heroku.com/>)
- Python installed

- GIT installed on local PC (on <https://git-scm.com/downloads>)
- A trained ML model developed in python (model used in Week4)
 - The shape of the dataset is 7796×3.
 - We built a **TfidfVectorizer** on our dataset using **sklearn**.
 - We used **PassiveAggressive Classifier**.
 - We saved the built model using **pickle**.

The picture below shows the trained model and the flask API.

```

1 import numpy as np
2 import pandas as pd
3 import pickle
4 from tensorflow import keras
5 from keras.preprocessing.sequence import pad_sequences
6 from keras.models import Sequential
7 from keras.layers.embeddings import Embedding
8 from sklearn.model_selection import train_test_split
9 from sklearn.feature_extraction.text import TfidfVectorizer
10 from sklearn.linear_model import PassiveAggressiveClassifier
11 from sklearn.metrics import accuracy_score, confusion_matrix
12 from keras import utils
13
14 df=pd.read_csv("news.csv")
15 df.head()
16
17
18 labels = df.label
19 X_train, X_test, y_train, y_test = train_test_split(df['text'],labels,test_size=1)
20
21 tfidf_vectorizer = TfidfVectorizer(stop_words='english', max_df=0.7)
22
23 # Fit and transform train set, transform test set
24 tfidf_train = tfidf_vectorizer.fit_transform(X_train)
25 tfidf_test = tfidf_vectorizer.transform(X_test)
26
27 # Initialize a PassiveAggressiveClassifier
28 pac = PassiveAggressiveClassifier(max_iter=50)
29 pac.fit(tfidf_train,y_train)
30
31 # saving vectorizer
32 with open('tfidf.pickle','wb') as f:
33     pickle.dump(tfidf_vectorizer,f)
34
35 # saving model
36 with open('model_fakenews.pickle','wb') as f:
37     pickle.dump(pac,f)

```

```

1 from flask import Flask, render_template, request, url_for, Markup, jsonify
2 import pickle
3
4 app = Flask(__name__)
5 pickle_in = open('model_fakenews.pickle', 'rb')
6 pac = pickle.load(pickle_in)
7 tfidf = open('tfidf.pickle', 'rb')
8 tfidf_vectorizer = pickle.load(tfidf)
9
10 @app.route('/')
11 def home():
12     return render_template("index.html")
13
14 @app.route('/newscheck')
15 def newscheck():
16     abc = request.args.get('news')
17     input_data = [abc.rstrip()]
18     # transforming input
19     tfidf_test = tfidf_vectorizer.transform(input_data)
20     # predicting the input
21     y_pred = pac.predict(tfidf_test)
22     return jsonify(result = y_pred[0])
23
24
25 if __name__ == '__main__':
26     app.run(debug=True)

```

- Prepare the required files

➤ create a file called **requirements.txt**

We should define first which libraries the application uses. That way, Heroku knows which ones to provide for us, similar to how we install them locally when developing the app.

To achieve this, we need to create a **requirements.txt** file with all of the modules:

```

dell@dell-PC MINGW64 /c/users/dell/desktop/Week4 (master)
$ py -m pip freeze > requirements.txt

```

This way we end up with a requirements.txt file that contains the libraries we're using and their versions:

```
attrs==21.2.0
certifi==2021.5.30
chardet==4.0.0
click==8.0.1
colorama==0.4.4
Cython==0.29.23
Flask==2.0.1
gunicorn==20.1.0
idna==2.10
importlib-metadata==4.4.0
iniconfig==1.1.1
itsdangerous==2.0.1
Jinja2==3.0.1
joblib==1.0.1
MarkupSafe==2.0.1
numpy==1.20.3
opencv-python==4.5.3.56
packaging==20.9
pluggy==0.13.1
py==1.10.0
pyparsing==2.4.7
pytest==6.2.4
python-dotenv==0.18.0
requests==2.25.1
scikit-learn==0.24.2
scipy==1.7.0
threadpoolctl==2.2.0
toml==0.10.2
typing-extensions==3.10.0.0
urllib3==1.26.6
Werkzeug==2.0.1
zipp==3.4.1
```

➤ Create a file called Procfile

For Heroku to be able to run our application like it should, we need to define a set of processes/commands that it should run beforehand.

```
dell@dell-PC MINGW64 /c/users/dell/desktop/Week4 (master)
$ echo web: gunicorn run:app>> Procfile
```

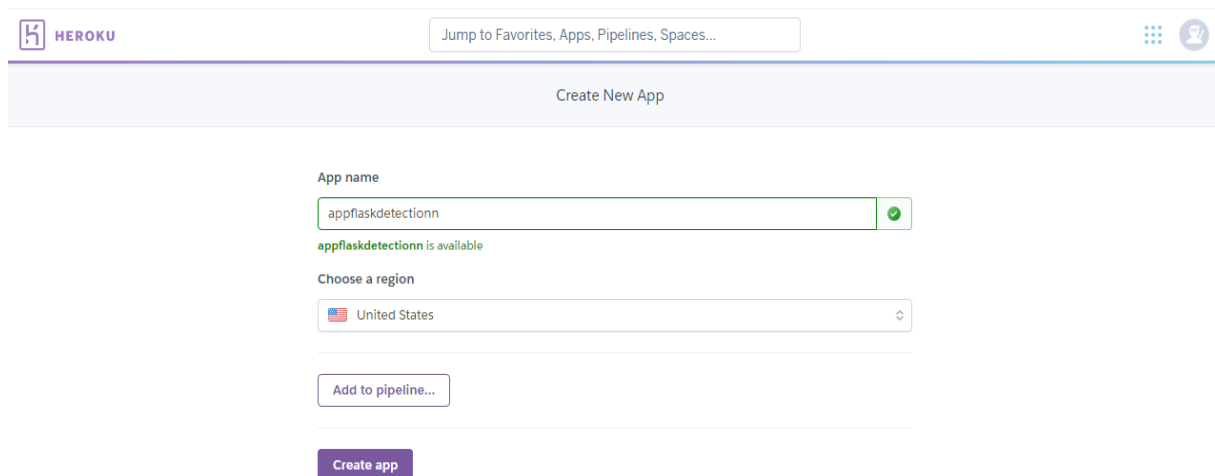
These commands are located in the **Procfile**:

```
web: gunicorn app:app
```

The web command tells Heroku to start a web server for the application, using gunicorn. Since our application is called app.py, we've set the app name to be app as well.

2. Delpoy the app to Heroku

First of all we sign up in heroku and creat an application from Heroku dashboard.



The screenshot shows the Heroku dashboard's 'Create New App' form. At the top, there's a Heroku logo and a search bar. Below the header, the title 'Create New App' is centered. The form contains the following elements: an 'App name' field with the value 'appflaskdetectionn' and a green checkmark icon, a message 'appflaskdetectionn is available', a 'Choose a region' dropdown menu with 'United States' selected, an 'Add to pipeline...' button, and a 'Create app' button at the bottom.

Then, connect to the app and choose a deployment method, which is in our case Github.

Deployment method



App connected to GitHub

Code diffs, manual and auto deploys are available for this app.

Connected to [jamilahHa/Flask-app-fake-news-detection](#) by [jamilahHa](#)

[Disconnect...](#)

Releases in the [activity feed](#) link to GitHub to view commit diffs

After choosing Github method deployment we deploy the app.

Deploy a GitHub branch

This will deploy the current state of the branch you specify below. [Learn more.](#)

Choose a branch to deploy

 heroku-deployment 



Deploy Branch

After clicking on deploy a “**Your app was successfully deployed**” message is displayed!

Deploy a GitHub branch

This will deploy the current state of the branch you specify below. [Learn more.](#)


Choose a branch to deploy

 heroku-deployment 

Deploy Branch

Receive code from GitHub 

Build **heroku-deployment** `aa07d621` 

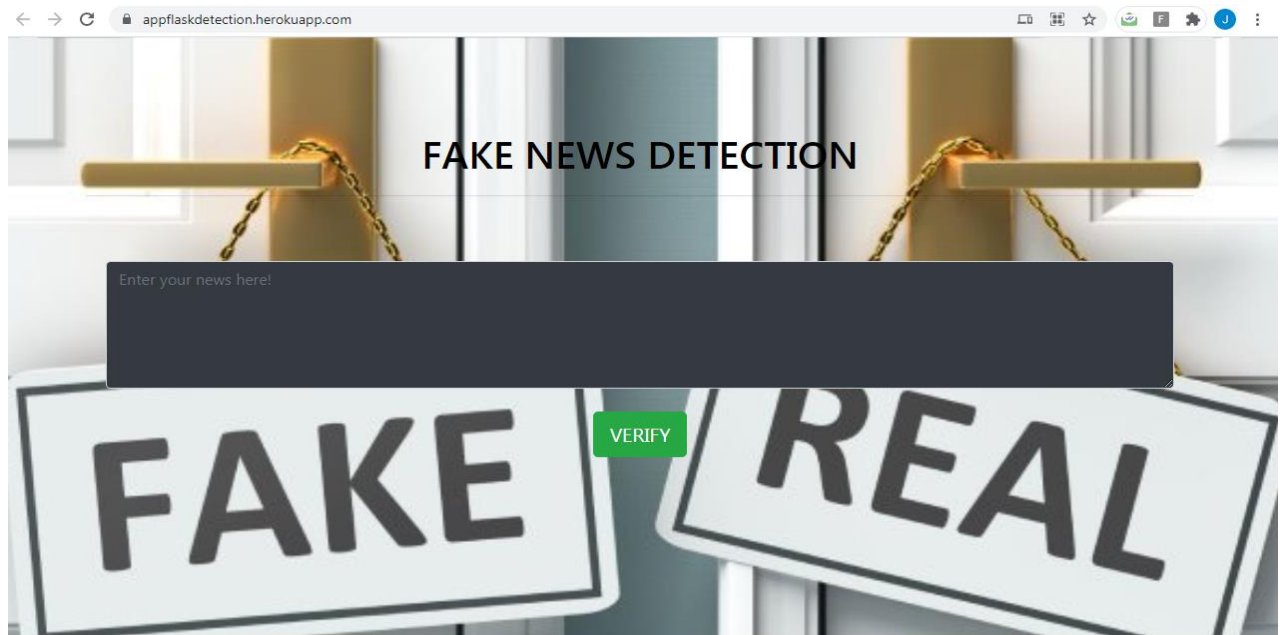
Release phase 

Deploy to Heroku 

Your app was successfully deployed.

 View

Now, the app is ready to use on <https://appflaskdetection.herokuapp.com/>



3. Testing the app

