Model Deployment using Flask

- Name: Mohammad Hani Bani Abed Alghani
- Batch code : LISUM32
- Submission date : 24/4/2024
- Submitted to: Data Glacier

Overview

- Deploying your basic machine learning model
- Learn how to use Flask to deploy a machine learning model into production
- Model deployment is a core topic in data scientist interviews – so start
- learning!

Abstract

This project has been written for the beginners of model deployment. With a

Nlp task it’s recommendation system using cosine similarity

Table of Contents:

What is Model Deployment?

Deployment is the method by which you integrate a machine learning model into an existing production environment to make practical business decisions based on data. In this way, we turn the model we have created into a product. At the same time, we offer the product to the user side.

What is Flask?

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies, and several common framework-related tools. The only feature that distinguishes Flask from other frameworks is that it is very easy to use.

Installing Flask on your Machine

Installing Flask is simple and straightforward. I generally use pip installed.

If you are using pip

$ pip install flask

Setting up the Project Workflow

- Build the Vectorizer
- Save the model and setup app
- Webpage Template
- Predict next songs and send results

Code

Importing Libraries

```
#Data Manipulation
#================================================
import numpy as np
import pandas as pd
#Regex
#================================================
import re
#NLTK
#================================================
import nltk
from nltk.stem.porter import PorterStemmer
#nltk.download('punkt')

#Vectorizer
#================================================
from sklearn.feature_extraction.text import TfidfVectorizer

#similarity_score
#================================================
from sklearn.metrics.pairwise import cosine_similarity

#Pickel
#================================================
import pickle
```

Read Data

```
df=pd.read_csv('E:/My_Project/CodeAlpha/Task 1/My_Data/spotify_millsongdata.csv')
```

```
df.head()
```

	artist	song	link	text
0	ABBA	Ahe's My Kind Of Girl	/a/abba/ahes+my+kind+of+girl_20598417.html	Look at her face, it's a wonderful face \r\nA...
1	ABBA	Andante, Andante	/a/abba/andante+andante_20002708.html	Take it easy with me, please \r\nTouch me gen...
2	ABBA	As Good As New	/a/abba/as+good+as+new_20003033.html	I'll never know why I had to go \r\nWhy I had...
3	ABBA	Bang	/a/abba/bang_20598415.html	Making somebody happy is a question of give an...
4	ABBA	Bang-A-Boomerang	/a/abba/bang+a+boomerang_20002668.html	Making somebody happy is a question of give an...

```
df.tail()
```

	artist	song	link	text
57645	Ziggy Marley	Good Old Days	/z/ziggy+marley/good+old+days_10198588.html	Irie days come on play \r\nLet the angels fly...
57646	Ziggy Marley	Hand To Mouth	/z/ziggy+marley/hand+to+mouth_20531167.html	Power to the workers \r\nMore power \r\nPowe...
57647	Zwan	Come With Me	/z/zwan/come+with+me_20148981.html	all you need \r\nis something i'll believe \...
57648	Zwan	Desire	/z/zwan/desire_20148986.html	northern star \r\nam i frightened \r\nwhere ...
57649	Zwan	Heartsong	/z/zwan/heartsong_20148991.html	come in \r\nmake yourself at home \r\ni'm a ...

```
df.shape
```

```
(57650, 4)
```

```
df.isnull().sum()
```

```
artist    0
song      0
link      0
text      0
dtype: int64
```

```
df=df.sample(10000).drop(columns=['link'],axis=1).reset_index(drop=True)
```

```
df.head()
```

	artist	song	text
0	Snoop Dogg	I'm Threw Witchu	[Snoop Dogg] \r\nYeah man, this is another 9 ...
1	Bruno Mars	Madly In Love With You	Madly In Love With You \r\n \r\nSee you down...
2	Chicago	Christmas Time Is Here	Christmas time is here \r\nHappiness and chee...
3	Boney M.	Young, Free And Single	Radio speaker: (not on album version) \r\nWel...
4	INXS	Doctor	We're all running fast tonight \r\nRocket sho...

```
df['text'][0]
```

```
"[Snoop Dogg]  \r\nYeah man, this is another 9 inch dick classic  \r\nWe want to dedicate this one, to all the fellas  \r\nWho got a hard head bitch, who just won't listen  \r\nWhy'know? The best thing
```

Text Preprocessing

Extract the Lower Case and apply some regex

```
df['text']=df['text'].str.lower().replace(r'^\w\s','').replace(r'\n',' ',regex=True)#\w : word \s : space
```

```
df.tail()
```

	artist	song	text
9995	Loretta Lynn	Get What'cha Got And Go	a pretty boy charlie's the name that you've be...
9996	Stevie Wonder	Part-Time Lover	call up, ring once, hang up the phone \r to l...
9997	Neil Young	Barstool Blues	if i could hold on \r to just one thought \r...
9998	Vince Gill	Don't Pretend With Me	they say you'll lie, they say you'll cheat \r...
9999	Weezer	In The Garage	i've got a dungeon master's guide \r i've got...

Steamming and Text Vectorization

```
stemmer=PorterStemmer()
```

```
def token(txt):
    token=nltk.word_tokenize(txt)
    a=[stemmer.stem(w) for w in token]
    return " ".join(a)
```

```
df['text']=df['text'].apply(lambda x:token(x))
```

```
tfid=TfidfVectorizer(analyzer='word',stop_words='english')
```

```
matrix=tfid.fit_transform(df['text'])
```

```
similar=cosine_similarity(matrix)
```

```
[104]
similar=cosine_similarity(matrix)
```

```
[105]
similar[0]
```

```
array([1.        , 0.02933386, 0.0091598 , ..., 0.03304134, 0.03433584,
       0.0660733 ])
```

+ Code + Markdown

```
[107]
df[df['song']=='Part-Time Lover'].index[0]
```

```
9996
```

Recommender Function

```
distance = sorted(
    list(enumerate(similar[idx])),
    reverse=True,
    key=lambda x: x[1]
)
```

The line of code calculates the similarity distance between the input song and all other songs in the dataset.

1. `similar` seems to be a 2D array or a list of lists, where `similar[idx]` retrieves the similarity scores of the input song with all other songs.

2. The `enumerate` function adds a counter to an iterable and returns it as an enumerate object. In this case, it enumerates over the similarity scores.

3. The `sorted` function then sorts these similarities in descending order. The `reverse=True` parameter indicates that the sorting should be in descending order.

4. The `key=lambda x: x[1]` parameter specifies that the sorting should be based on the second element of each tuple within the list (which represents the similarity score).

```
[108]
def recommender(song_name):
    idx=df[df['song']==song_name].index[0] #### check the id of each song
    distance=sorted(list(enumerate(similar[idx])) , reverse=True, key=lambda x:x[1]) #### Distance Calculation
    song=[]
    for s_id in distance[1:5]:     ### see the top 5
        song.append(df.iloc[s_id[0]].song)
    return song
```

```
[109]
recommender('Part-Time Lover')
```

```
['Lover Come Back', 'Like Lovers Do', 'No Money Down', 'Lovers On The Sun']
```

```
[111]
pickle.dump(similar,open("similarity",'wb'))
```

```
pickle.dump(df,open("df",'wb'))
```

Flask Code

```python
import pickle
from flask import Flask, render_template, request
import spotipy
from spotipy.oauth2 import SpotifyClientCredentials

app = Flask(__name__)

# Initialize Spotipy client
Client_id = 'f21e8dc62f2e40ea923b8d67e047542c'
Client_secret = '276d6f78a925465f9849cbe168ec1142'
client_credentials_manager = SpotifyClientCredentials(client_id=Client_id, client_secret=Client_secret)
sp = spotipy.Spotify(client_credentials_manager=client_credentials_manager)

# Load your data and models
music = pickle.load(open('E:/My_Project/CodeAlpha/Task 1/df', 'rb'))
similarity = pickle.load(open('E:/My_Project/CodeAlpha/Task 1/similarity', 'rb'))

def get_song_album_cover_url(song_name, artist_name):
    search_query = f"track:{song_name} artist:{artist_name}"
    results = sp.search(q=search_query, type='track')

    if results and results['tracks']['items']:
        track = results['tracks']['items'][0]
        album_cover_url = track['album']['images'][0]['url']
        return album_cover_url
    else:
        return "https://i.postimg.cc/0QNxYz4V/social.png"

def recommender(song):
    idx = music[music['song'] == song].index[0]
    distance = sorted(list(enumerate(similarity[idx])), reverse=True, key=lambda x: x[1])

    recommened_music_name = []
    recommended_music_poster = []
    for i in distance[1:6]:
        artist = music.iloc[i[0]]['artist']
        recommended_music_poster.append(get_song_album_cover_url(music.iloc[i[0]]['song'], artist))
        recommened_music_name.append(music.iloc[i[0]]['song'])
    return recommened_music_name, recommended_music_poster

@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        selected_song = request.form['song']
        recommended_music_names, recommended_music_posters = recommender(selected_song)

        # Pre-zip the lists
        recommendations = zip(recommended_music_names, recommended_music_posters)
```

```python
def get_song_album_cover_url(song_name, artist_name):
    search_query = f"track:{song_name} artist:{artist_name}"
    results = sp.search(q=search_query, type='track')

    if results and results['tracks']['items']:
        track = results['tracks']['items'][0]
        album_cover_url = track['album']['images'][0]['url']
        return album_cover_url
    else:
        return "https://i.postimg.cc/0QNxYz4V/social.png"

def recommender(song):
    idx = music[music['song'] == song].index[0]
    distance = sorted(list(enumerate(similarity[idx])), reverse=True, key=lambda x: x[1])

    recommened_music_name = []
    recommended_music_poster = []
    for i in distance[1:6]:
        artist = music.iloc[i[0]]['artist']
        recommended_music_poster.append(get_song_album_cover_url(music.iloc[i[0]]['song'], artist))
        recommened_music_name.append(music.iloc[i[0]]['song'])
    return recommened_music_name, recommended_music_poster

@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        selected_song = request.form['song']
        recommended_music_names, recommended_music_posters = recommender(selected_song)

        # Pre-zip the lists
        recommendations = zip(recommended_music_names, recommended_music_posters)

        return render_template('index.html',
                               songs=music['song'].values,
                               selected_song=selected_song,
                               recommendations=recommendations)
    else:
        return render_template('index.html', songs=music['song'].values)


if __name__ == '__main__':
    app.run(debug=True)
```

HTML CODE :

```html
<!DOCTYPE html>
<html>
<head>
    <title>Song Recommender</title>
    <style>
        body {
            font-family: 'Helvetica Neue', Helvetica, Arial, sans-serif;
            background-color: #1DB954;
            color: #ffffff;
            margin: 0;
            padding: 0;
        }
        h1 {
            font-size: 36px;
            margin-top: 30px;
            margin-bottom: 30px;
            text-align: center;
        }
        form {
            text-align: center;
            margin-bottom: 20px;
        }
        label {
            font-size: 18px;
        }
        select {
            font-size: 16px;
            padding: 10px;
            border: none;
            border-radius: 5px;
            background-color: #ffffff;
            color: #000000;
        }
        input[type="submit"] {
            font-size: 16px;
            padding: 10px 20px;
            border: none;
            border-radius: 5px;
            background-color: #1ED760;
            color: #ffffff;
            cursor: pointer;
        }
        .recommendations {
            display: flex;
            flex-wrap: wrap;
            justify-content: center;
        }
```

```
        .recommendation {
            margin: 10px;
            text-align: center;
        }
        .recommendation p {
            font-size: 16px;
            margin-top: 10px;
        }
        .recommendation img {
            width: 200px;
            height: 200px;
            border-radius: 10px;
            margin-top: 10px;
        }
    </style>
</head>
<body>
    <h1>Song Recommender</h1>
    <form action="/" method="post">
        <label for="song">Select a song:</label><br>
        <select id="song" name="song">
            {% for song in songs %}
            <option value="{{ song }}">{{ song }}</option>
            {% endfor %}
        </select><br><br>
        <input type="submit" value="Show Recommendation">
    </form>

    {% if selected_song %}
    <h2>Recommendations for {{ selected_song }}</h2>
    <div class="recommendations">
        {% for name, poster in recommendations %}
            <div class="recommendation">
                <p>{{ name }}</p>
                <img src="{{ poster }}" alt="Album Cover">
            </div>
        {% endfor %}
    </div>
{% endif %}
</body>
</html>
```

Song Recommender

Select a song:

Christmas Time Is Here

Show Recommendation

Recommendations for Christmas Time Is Here

Christmas Time Is Here	Christmas Time	This Time Of Year	This Christmas	This Christmas