



Data Science Internship at Data Glacier

Week 5: Cloud and API Deployment

Name: Amrapali Mhaisgawali

Batch code: LISUM14

Submission date: 03/11/2022

Submitted to: Data Glacier

Index:

1.	Introduction	3
2.	Problem Statement	3
	2.1 Dataset Used	3
	2.2 Data Information	3
	2.3 Attribute Used	4
	2.4 Application Workflow	4
3.	Building Machine Learning Model	5
	3.1 Importing required libraries and dataset	5
	3.2 Data pre-processing	5
	3.3 Model Creation (support vector machine.jpynb).....	6
	3.4 Save the model	6
4.	Deployment of Model into Flask Framework	6
	4.1 App.py	7
	4.2 Home.html	8
	4.3 Styles.css	9
	4.4 Result.html	10
	4.5 Results with Running Procedure	10
5	Model development using Heroku	12
	4.1 Steps for Model Development using Heroku	12
6	References	15

1. Introduction

YouTube is one of the biggest site for user get information on the Internet [1]. Because of that, many spammers will trick the YouTube user by spamming the YouTube comments. According to Hamou [2], spam is now a trend attack and the YouTube defines spam as inappropriate comments, such as abuse or trolling and also people trying to sell things. Ham can be defined as “good comments” or YouTube free from spam comment. Spam can be categorized as dangerous because spam has the potential of cyber security threat for end users. The spammer used this opportunity to spread malware through comment fields, which will exploit vulnerabilities in the user’s machines. Another intention includes seizing money transactions and hijacking credit card and banking information. Besides, spammer tends to ruin the content of web pages. This action will lead visitors to annoy overall of the posted content [3].

2. Problem Statement

YouTube has its own spam filtering system, though there are still spam comments that are not being caught[4]. So the is developed for detecting spam from using dataset of youtube comments.

2.1 Dataset Used

The dataset used is Youtube spam collection[5] for spam detection. It is a public set of comments collected for spam research. It has five datasets composed by 1,956 real messages extracted from five videos that were among the 10 most viewed on the collection period.

2.2 Data Information

The samples were extracted from the comments section of five videos that were among the 10 most viewed on YouTube during the collection period. The table below lists the datasets, the YouTube video ID, the number of samples in each class and the total number of samples per dataset.

Dataset	YouTube ID	Spam	Ham	Total
Psy	9bZkp7q19f0	175	175	350
KatyPerry	CevxZvSJLk8	175	175	350
LMFAO	KQ6zr6kCPj8	236	202	438
Eminem	uelHwf8o7_U	245	203	448
Shakira	pRpeEdMmmQ0	174	196	370

Table 1: Dataset Information

2.3 Attribute Information:

The collection is composed by one CSV file per dataset, where each line has the following attributes:

Attribute	Meaning
COMMENT_ID	Unique ID of Comment
AUTHOR	Name of author who posted comment
DATE	Actual date of comment posted
CONTENT	Comment given by author
CLASS	Class of comment 1-spam, 0-not spam

Table 2: Attribute Information

2.4 Application Workflow

In this, SVM machine learning model is used and the Flask Framework for the deployment. As a demonstration, model will help to predict the spam and ham comment given by user.

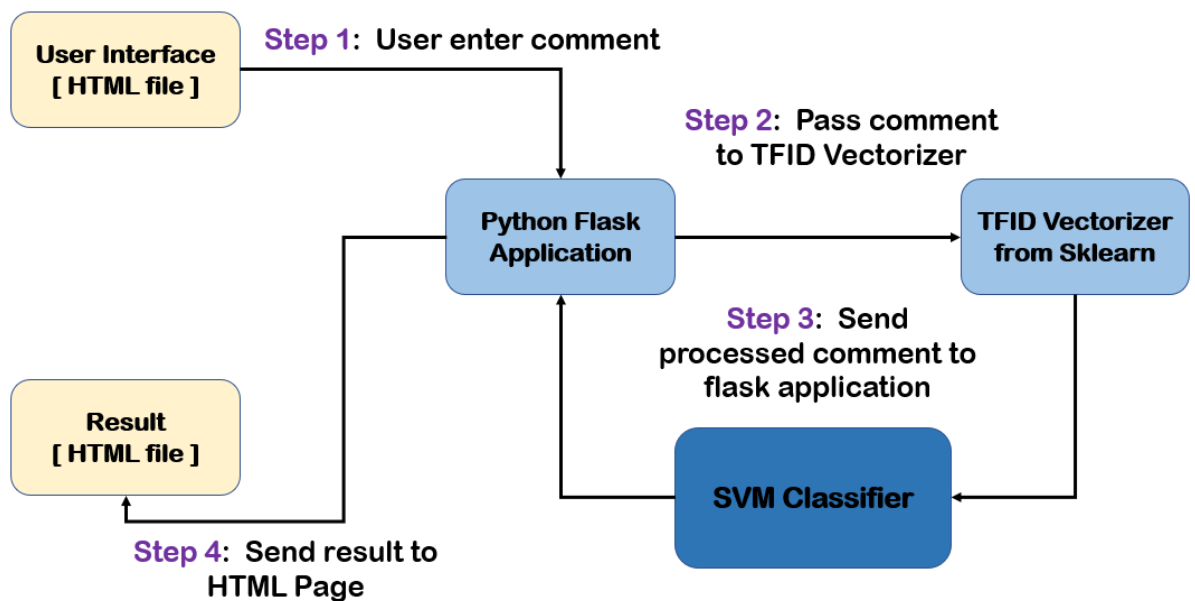


Fig. 2.1 Application Workflow

A machine learning model is built for YouTube comments from five different videos, then creates an API for the model using Flask Framework and python micro-

framework for building web application. This API allows to predictive capabilities through HTTP requests.

3. Building Machine Learning Model (support vector machine.jpynb)

3.1 Import required libraries and dataset

In this required libraries and comment from five different videos are imported for model creation.

Part 1: Import Libraries and Dataset

```
In [1]: # import Libraries & Packages
import numpy as np          # Import Numpy for data statistical analysis
import pandas as pd         # Import Pandas for data manipulation using dataframes
import seaborn as sns       # Statistical data visualization
import matplotlib.pyplot as plt # Import matplotlib for data visualisation

executed in 3.25s, finished 15:23:16 2022-10-26
```

```
In [2]: # Import Youtube Ham or Spam dataset taken from UCI
df1 = pd.read_csv("dataset/Youtube01-Psy.csv")          # Psy youtube channel most viewed video comments dataset
df2 = pd.read_csv("dataset/Youtube02-KatyPerry.csv")    # KatyPerry youtube channel most viewed video comments dataset
df3 = pd.read_csv("dataset/Youtube03-LMFAO.csv")        # Psy LMFAO channel most viewed video comments dataset
df4 = pd.read_csv("dataset/Youtube04-Eminem.csv")       # Eminem youtube channel most viewed video comments dataset
df5 = pd.read_csv("dataset/Youtube05-Shakira.csv")      # Shakira youtube channel most viewed video comments dataset

executed in 90ms, finished 15:23:17 2022-10-26
```

```
In [3]: # Merge all the dataset into single file
frames = [df1, df2, df3, df4, df5]                    # make a list of all file
df_merged = pd.concat(frames)                         # concatenate the all the file into single
keys = ["Psy", "KatyPerry", "LMFAO", "Eminem", "Shakira"] # Merging with Keys
df_with_keys = pd.concat(frames, keys=keys)            # concatenate data with keys
dataset = df_with_keys

executed in 13ms, finished 15:23:18 2022-10-26
```

```
In [4]: # Information about dataset
print(dataset.size)          # size of dataset
print(dataset.shape)         # shape of dataset
print(dataset.keys())        # attributes of dataset

executed in 6ms, finished 15:23:19 2022-10-26
```

9780
(1956, 5)
Index(['COMMENT_ID', 'AUTHOR', 'DATE', 'CONTENT', 'CLASS'], dtype='object')

3.2 Data Pre-processing

For creating a predictive model dataset is split into 80% training and 20% testing. A Term Frequency-Inverse document frequency (TF-IDF) vectorizer is used to transform the words into numerical features (numpy arrays) for training and testing purpose.

Part 2: Data Preprocessing

```
In [5]: # working with text content
dataset = dataset[['CONTENT', 'CLASS']]                # context = comments of viewers & Class = ham or Spam

executed in 5ms, finished 15:23:22 2022-10-26
```

```
In [6]: # Predictor and Target attribute
dataset_X = dataset['CONTENT']                        # predictor attribute
dataset_y = dataset['CLASS']                         # target attribute

executed in 4ms, finished 15:23:23 2022-10-26
```

```
In [7]: # Feature Extraction from Text using TF-IDF model
from sklearn.feature_extraction.text import TfidfVectorizer # import TF-IDF model from scikit Learn

executed in 288ms, finished 15:23:24 2022-10-26
```

```
In [8]: # Extract Feature With TF-IDF model
corpus = dataset_X                                   # declare the variable
cv = TfidfVectorizer()                              # initialize the TF-IDF model
X = cv.fit_transform(corpus).toarray()               # fit the corpus data into BOW model

executed in 79ms, finished 15:23:25 2022-10-26
```

```
In [9]: # Split the dataset into Train and Test
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, dataset_y, test_size=0.2, random_state=0)

executed in 147ms, finished 15:23:26 2022-10-26
```

3.3 Model Creation

After data pre-processing, a machine learning model is created to classify the YouTube spam comments. For this purpose, Support Vector Machine (SVM) algorithm is used from scikit-learn. After importing and initialize SVM model the dataset is being fitted for training using classifier.

Part 3: Building a Model

```
In [11]: # import the model from sklearn
from sklearn.svm import SVC          # import the Support Vector Machine Classifier model
executed in 156ms, finished 15:23:28 2022-10-26

In [12]: # initialize the model
classifier = SVC(kernel = 'linear', random_state= 0)
executed in 4ms, finished 15:23:29 2022-10-26

In [13]: # fit the dataset into our classifier model for training
classifier.fit(X_train, y_train)
executed in 2.90s, finished 15:23:32 2022-10-26

Out[13]: SVC(kernel='linear', random_state=0)
```

3.4 Save the model

Last step is saving the model using pickle.

```
In [17]: # import pickle library
import pickle          # pickle used for serializing and de-serializing a Python object structure
executed in 4ms, finished 15:23:43 2022-10-26

In [18]: # save the model (Serialization using pickle)
Support_Vector_Machine = open("model.pkl", "wb")          # open the file for writing
pickle.dump(classifier, Support_Vector_Machine)            # dumps an object to a file object
Support_Vector_Machine.close()                             # here we close the fileObject
executed in 38ms, finished 15:23:44 2022-10-26
```

4. Deployment of model into flask framework

A web application is developed that consists of a two-web pages, one with a form field that lets us enter a comment. After submitting the comment to the web application, it will redirect it on a result page which gives us a result of spam or ham (not spam).

Following is the directory structure of all files used for application.

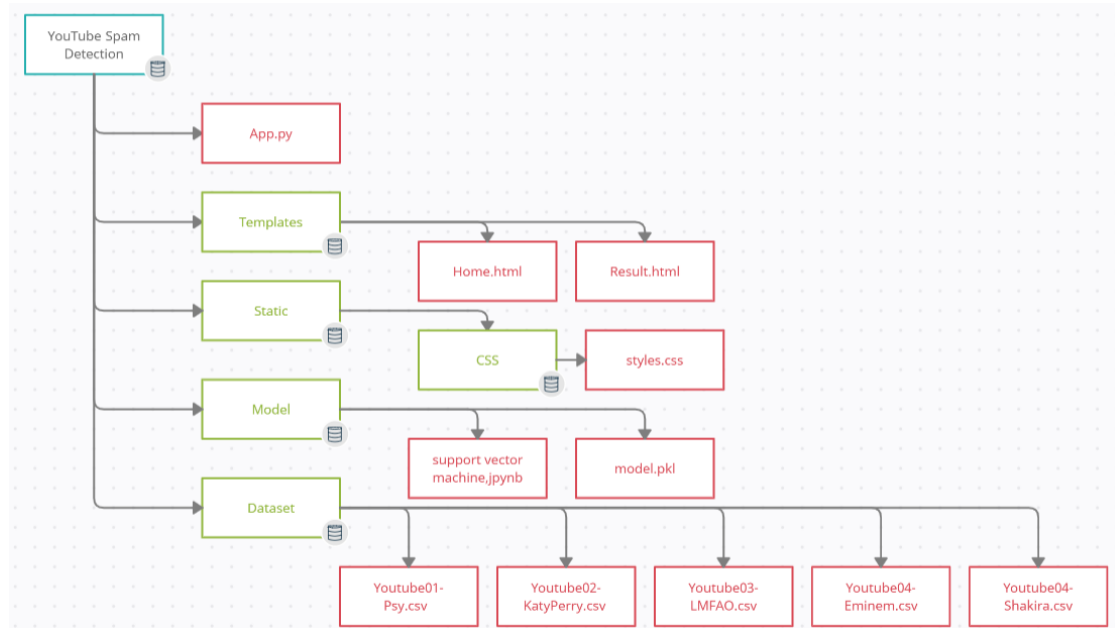


Fig 4.1 Directory Structure

4.1 App.py

The app.py file contains the source code including the ML code for classification and will be executed by the Python interpreter to run the Flask web application.

```

1 from flask import Flask,render_template,url_for,request
2 from sklearn.feature_extraction.text import TfidfVectorizer
3 import pandas as pd
4 import pickle
5
6 app = Flask(__name__)
7
8 @app.route('/')
9 def home():
10     return render_template('home.html')
11
12 @app.route('/predict',methods=['POST'])
13 def predict():
14     df1 = pd.read_csv("dataset/Youtube01-Psy.csv")           # Psy youtube channel most viewed video comments dataset
15     df2 = pd.read_csv("dataset/Youtube02-KatyPerry.csv")      # KatyPerry youtube channel most viewed video comments dataset
16     df3 = pd.read_csv("dataset/Youtube03-LMFAO.csv")          # Psy LMFAO channel most viewed video comments dataset
17     df4 = pd.read_csv("dataset/Youtube04-Eminem.csv")         # Eminem youtube channel most viewed video comments dataset
18     df5 = pd.read_csv("dataset/Youtube05-Shakira.csv")        # Shakira youtube channel most viewed video comments dataset
19
20     # Merge all the dataset into single file
21     frames = [df1,df2,df3,df4,df5]                           # make a List of all file
22     df_merged = pd.concat(frames)                             # concatenate the all the file into single
23     keys = ["Psy","KatyPerry","LMFAO","Eminem","Shakira"]     # Merging with Keys
24     df_with_keys = pd.concat(frames,keys=keys)                # concatenate data with keys
25     dataset=df_with_keys
26
27     # working with text content
28     dataset = dataset[["CONTENT" , "CLASS"]]                  # context = comments of viewers & Class = ham or Spam
29
30     # Predictor and Target attribute
31     dataset_X = dataset['CONTENT']                            # predictor attribute
32     dataset_y = dataset['CLASS']                             # target attribute
33
34     # Extract Feature With TF-IDF model
35     corpus = dataset_X                                         # declare the variable
36     cv = TfidfVectorizer()                                     # initialize the TF-IDF model
37     X = cv.fit_transform(corpus).toarray()                    # fit the corpus data into BOW model
38
39
40     # import pickle file of my model
41     model = open("model/model.pkl","rb")
42     clf = pickle.load(model)
43
44     if request.method == 'POST':
45         comment = request.form['comment']
46         data = [comment]
47         vect = cv.transform(data).toarray()
48         my_prediction = clf.predict(vect)
49         return render_template('result.html',prediction = my_prediction)
50
51
52 if __name__ == '__main__':
53     app.run(debug=True)
  
```

- Application will run as a single module; thus, a new Flask instance is initialized with the argument `__name__` to let Flask know that it can find the HTML template folder (templates) in the same directory where it is located.
- Next, the route decorator is used (`@app.route('/')`) to specify the URL that should trigger the execution of the home function. Home function simply rendered the `home.html` HTML file, which is in the templates folder.
- Predict function has the spam data set, it pre-processes the text, and make predictions, and then store the model. A new comment is entered by the user and uses the model to make a prediction for its label.
- The POST method is used to transport the form data to the server in the message body. Finally, by setting the `debug=True` argument inside the `app.run` method, it also activated Flask's debugger.
- The `run` function is used to only run the application on the server when this script is directly executed by the Python interpreter, which we ensured using the `if` statement with `__name__ == '__main__'`.

4.1 Home.html

The `home.html` file will render a text form where a user can enter a comment.

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Home</title>
5   <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='css/styles.css') }}">
6 </head>
7 <body>
8   <header>
9
10    <div class="container" >
11
12      <h2>Youtube Comments Spam Detection</h2>
13
14    </div>
15  </header>
16
17  <div class="ml-container">
18
19    <form action="{{ url_for('predict') }}" method="POST">
20      <p>Enter Your Comment Here</p>
21      <textarea name="comment" rows="4" cols="50"></textarea>
22      <br/>
23
24      <input type="submit" class="btn-info" value="predict">
25
26    </form>
27
28  </div>
29 </body>
30 </html>

```


4.2 Styles.css

CSS is to determine how the look and feel of HTML documents. styles.css must be saved in a sub-directory called static, which is the default directory where Flask looks for static files such as CSS.

```
1 body{
2     font:15px/1.5 Arial, Helvetica,sans-serif;
3     padding: 0px;
4     background-color:#f4f3f3;
5 }
6 .container{
7     width:100%;
8     margin: auto;
9     overflow: hidden;
10 }
11 header{
12     background:#03A9F4;#35434a;
13     border-bottom:#448AFF 3px solid;
14     height:120px;
15     width:100%;
16     padding-top:30px;
17 }
18 .main-header{
19     text-align:center;
20     background-color: blue;
21     height:100px;
22     width:100%;
23     margin:0px;
24 }
25 #brandname{
26     float:left;
27     font-size:30px;
28     color: #fff;
29     margin: 10px;
30 }
31 header h2{
32     text-align:center;
33     color:#fff;
34 }
35
36 .btn-info {background-color: #2196F3;
37     height:40px;
38     width:100px;} /* Blue */
39 .btn-info:hover {background: #0b7dda;}
40
41
42 .resultss{
43     border-radius: 15px 50px;
44     background: #345fe4;
45     padding: 20px;
46     width: 200px;
47     height: 150px;
48 }
```

4.4 Result.html

Result.html file will be rendered via the `render_template('result.html', prediction=my_prediction)` which is inside the predict function of `app.py` script to display the text that a user-submitted via the text field. Result.html has Jinja syntax : `{% if prediction == 1%},{% elif prediction == 0%},{% endif %}`, and is used to access the prediction returned from HTTP request within the HTML file.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <title></title>
5     <link rel="stylesheet" type="text/css" href="{ url_for('static', filename='css/styles.css') }">
6 </head>
7 <body>
8
9     <header>
10         <div class="container">
11             <h2>YouTube Comments Spam Detection</h2>
12
13         </div>
14     </header>
15     <p style="color:black;font-size:20;text-align:center;"><b>Results for Comment</b></p>
16     <div class="results">
17
18         {% if prediction == 1%}
19         <h2 style="color:red;">Spam</h2>
20         {% elif prediction == 0%}
21         <h2 style="color:green;">Not a Spam (It is a Ham)</h2>
22         {% endif %}
23
24     </div>
25
26 </body>
27 </html>
```

4.5 Results with running procedure


Following is a command to run `app.py`

```
(app) C:\Users\AMRAPALI\Documents\internship\Week4\ML web applicationusing Flask>python app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 875-509-621
```

Now open a web browser and navigate to <http://127.0.0.1:5000/> following is output of home.html.

Youtube Comments Spam Detection


Enter Your Comment Here



Enter a comment and click the Predict button

Youtube Comments Spam Detection

Enter Your Comment Here

Please subscribe to a channel

predict

Following is output which is given by result.html

YouTube Comments Spam Detection

Results for Comment

Spam

5. Model Development using Heroku

Once the model has been trained and tested locally, now it is ready for deploy on Heroku. There are a few ways to upload the application source code onto Heroku. The easiest way is to link a GitHub repository to Heroku account.

Requirement.txt It is a text file containing the python packages required to execute the application.

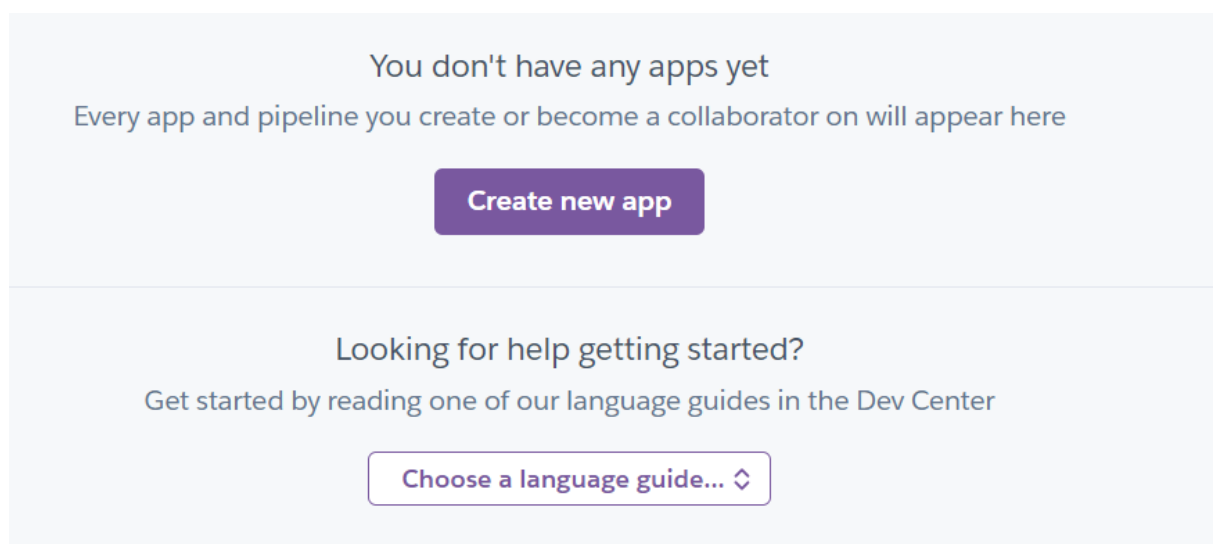
```
1 Flask==2.1.0
2 gunicorn==20.0.4
3 pandas==1.3.4
4 scikit-learn==0.24.2
5 numpy==1.23.4
6 sklearn==0.0
```

5.1 Steps for Model Deployment Using Heroku

Once files are uploaded to the GitHub repository, application is ready for deployment on Heroku.

Follow the steps below:

1. After sign up on heroku.com then click on Create new app.



2. Enter App name and region

App name

youtube-comment-detection-app1



youtube-comment-detection-app1 is available

Choose a region



United States



Add to pipeline...

Create app

3. Connect to GitHub repository where code is uploaded.

Deployment method



Heroku Git
Use Heroku CLI



GitHub
Connected



Container Registry
Use Heroku CLI

After that choose the repository where code is uploaded.

The screenshot shows a GitHub repository page for 'amrapali10 / Cloud-and-API-Deployment'. The repository is public and has 12 commits. The file list includes: dataset, model, static/css, templates, Profile, README.md, app.py, model.pkl, and requirements.txt. The README.md file is selected, showing the title 'Cloud-and-API-Deployment'. The right sidebar shows repository statistics: 0 stars, 1 watching, 0 forks. There are no releases or packages published. The environment 'youtube-comment-detection-app' is listed as active.

4. Deploy Branch

branch is always in a deployable state and any tests have passed before you push. [Learn more.](#)

Choose a branch to deploy

🔗 main

☐ Wait for CI to pass before deploy

Only enable this option if you have a Continuous Integration service configured on your repo.

Enable Automatic Deploys

Manual deploy

Deploy the current state of a branch to this app.

Deploy a GitHub branch

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

Choose a branch to deploy

🔗 main

Deploy Branch

5. After waiting 5 to 15 minutes application is Ready to run

Manual deploy

Deploy the current state of a branch to this app.

Deploy a GitHub branch

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

Choose a branch to deploy

🔗 master

Deploy Branch

Receive code from GitHub

✓

Build master 3a3e6e70

✓

Release phase

✓

Deploy to Heroku

✓

Your app was successfully deployed.

➡ View

The app is published at

<https://youtube-comment-detection-app.herokuapp.com/>

References:

- [1] Scheltus, P., Dorner, V., & Lehner, F. (2013). Leave a Comment! An In-Depth Analysis of User Comments on YouTube. *Wirtschaftsinformatik*, 42.
- [2] Hamou, R. M., Amine, A., & Tahar, M. (2017). The Impact of the Mode of Data Representation for the Result Quality of the Detection and Filtering of Spam. In *Ontologies and Big Data Considerations for Effective Intelligence*(pp. 150-168). IGI Global.
- [3] Alsaleh, M., Alarifi, A., Al-Quayed, F., & Al-Salman, A. (2016). Combating comment spam with machine learning approaches. *Proceedings - 2015 IEEE 14th International Conference on Machine Learning and Applications, ICMLA 2015*, 295–300. <https://doi.org/10.1109/ICMLA.2015.192>
- [4] H. Oh, "A YouTube Spam Comments Detection Scheme Using Cascaded Ensemble Machine Learning Model," in *IEEE Access*, vol. 9, pp. 144121-144128, 2021, doi: 10.1109/ACCESS.2021.3121508.
- [5] <https://archive.ics.uci.edu/ml/datasets/YouTube+Spam+Collection>