Data Glacier Internship

Week 6: Cloud and API deployment

Submitted by: Nahari Terena

Batch Code: LISUM15

Date: December, 2022

Introduction

Using the Flask framework, this project deploys machine learning model (SVM). The model predicts the spam or the ham comment on youtube's videos.

In this project, we focus on two main points: building the model in Python, then create an API for the model, using Flask.

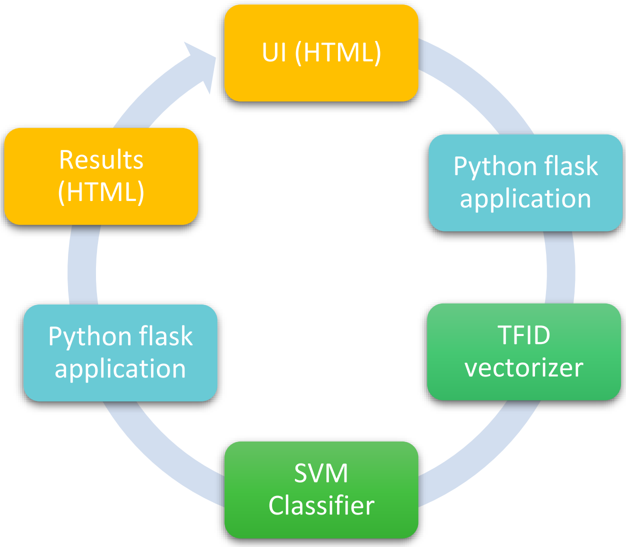


Figure 1.Application Workflow

Dataset Information

The dataset was donated to UCI Machine Learning repository in March 2017. It is a public set of comments collected for spam research. Although 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, we only use three of them: Katy Perry, Eminem, and Shakira because they are favorite ones of this author.

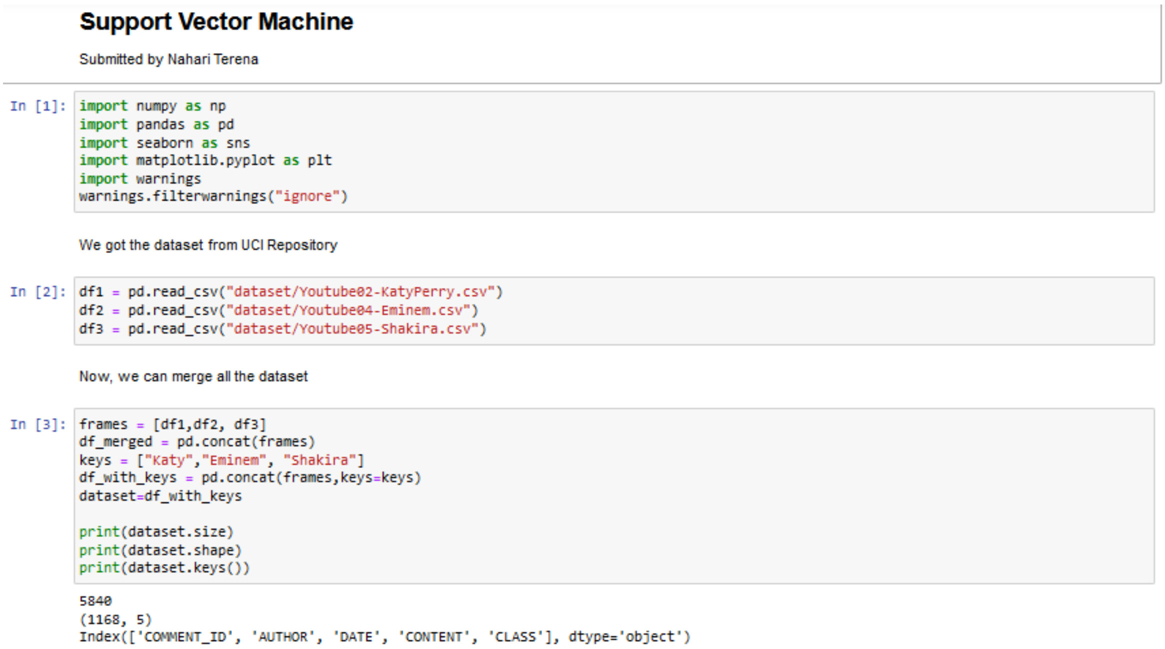
Dataset	ID	Spam	Ham	Total Comments
Psy	9bZkp7q19f0	175	175	350

Katy Perry	CevxZvSJLk8	175	175	350
LMFAO	KQ6zr6kCPj8	236	202	438
Eminem	uelHwf8o7_U	245	203	448
Shakira	pRpeEdMmmQ0	174	196	370

Each dataset has five attributes which includes: ID, author, date, content and class.

Building the Model

In the first part, we import some libraries necessary to aim the objective. Also, we checked the dataset and all the information read.



Support Vector Machine

Submitted by Nahari Terena

```
In [1]: import numpy as np
        import pandas as pd
        import seaborn as sns
        import matplotlib.pyplot as plt
        import warnings
        warnings.filterwarnings("ignore")
```

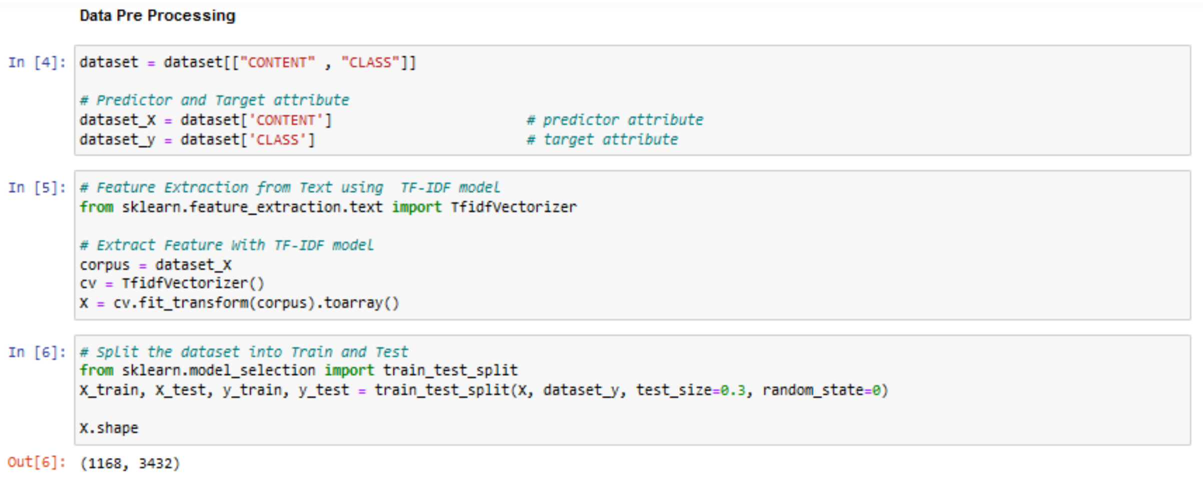
We got the dataset from UCI Repository

```
In [2]: df1 = pd.read_csv("dataset/Youtube02-KatyPerry.csv")
        df2 = pd.read_csv("dataset/Youtube04-Eminem.csv")
        df3 = pd.read_csv("dataset/Youtube05-Shakira.csv")
```

Now, we can merge all the dataset

```
In [3]: frames = [df1,df2, df3]
        df_merged = pd.concat(frames)
        keys = ["Katy","Eminem", "Shakira"]
        df_with_keys = pd.concat(frames,keys=keys)
        dataset=df_with_keys

        print(dataset.size)
        print(dataset.shape)
        print(dataset.keys())

        5840
        (1168, 5)
        Index(['COMMENT_ID', 'AUTHOR', 'DATE', 'CONTENT', 'CLASS'], dtype='object')
```

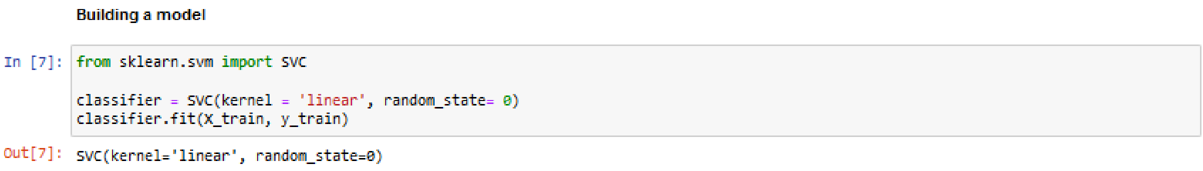
The dataset was split into two other groups: 30% for the test set and 70% for the training one. We fed our dataset into a Term Frequency-Inverse document frequency vectorizer (TF-IDF) which transforms words into numerical features for the two new datasets.



Data Pre Processing

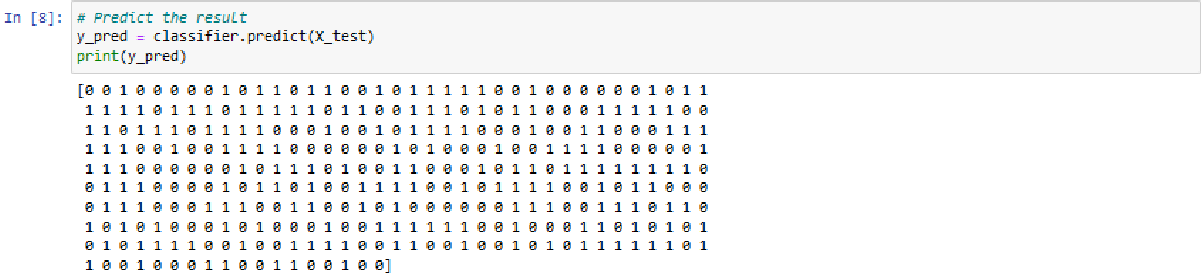
```
In [4]: dataset = dataset[["CONTENT" , "CLASS"]]

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

In [5]: # Feature Extraction from Text using  TF-IDF model
        from sklearn.feature_extraction.text import TfidfVectorizer

        # Extract Feature With TF-IDF model
        corpus = dataset_X
        cv = TfidfVectorizer()
        X = cv.fit_transform(corpus).toarray()

In [6]: # 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=0.3, random_state=0)

        X.shape

Out[6]: (1168, 3432)
```

Now, we may properly implement the machine learning model to classify each comment. For this purpose, we implement Support Vector Machine (SVM) using sckit-learn. Then, fit it into training dataset.



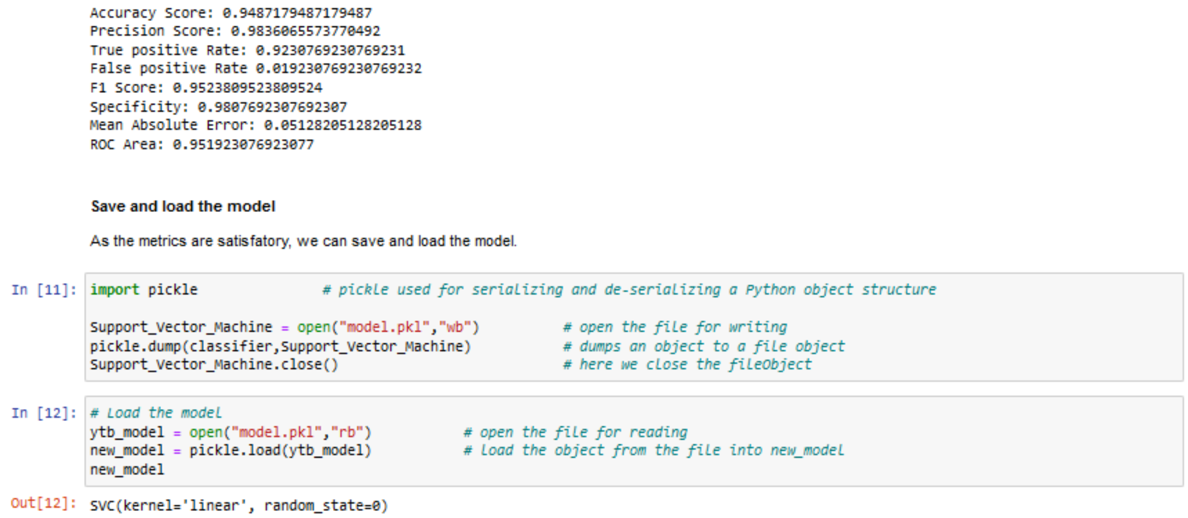
Building a model

```
In [7]: from sklearn.svm import SVC

        classifier = SVC(kernel = 'linear', random_state= 0)
        classifier.fit(X_train, y_train)

Out[7]: SVC(kernel='linear', random_state=0)
```



```
In [8]: # Predict the result
        y_pred = classifier.predict(X_test)
        print(y_pred)

        [0 0 1 0 0 0 0 0 1 0 1 1 0 1 1 0 0 1 0 1 1 1 1 1 0 0 1 0 0 0 0 0 0 1 0 1 1
         1 1 1 1 0 1 1 1 0 1 1 1 1 1 0 1 1 0 0 1 1 1 0 1 0 1 1 0 0 0 1 1 1 1 1 0 0
         1 1 0 1 1 1 0 1 1 1 1 0 0 0 1 0 0 1 0 1 1 1 1 0 0 0 1 0 0 1 1 0 0 0 1 1 1
         1 1 1 0 0 1 0 0 1 1 1 1 0 0 0 0 0 0 1 0 1 0 0 0 1 0 0 1 1 1 1 0 0 0 0 0 1
         1 1 1 0 0 0 0 0 0 1 0 1 1 1 0 1 0 0 1 1 0 0 0 1 0 1 1 0 1 1 1 1 1 1 1 1 0
         0 1 1 1 0 0 0 0 1 0 1 1 0 1 0 0 1 1 1 1 0 0 1 0 1 1 1 1 0 0 1 0 1 1 0 0 0
         0 1 1 1 0 0 0 1 1 1 0 0 1 1 0 0 1 0 1 0 0 0 0 0 0 1 1 1 0 0 1 1 1 0 1 1 0
         1 0 1 0 1 0 0 0 1 0 1 0 0 0 1 0 0 1 1 1 1 1 1 0 0 1 0 0 0 1 1 0 1 0 1 0 1
         0 1 0 1 1 1 1 0 0 1 0 0 1 1 1 1 0 0 1 1 0 0 1 0 0 1 0 1 0 1 1 1 1 1 1 0 1
         1 0 0 1 0 0 0 1 1 0 0 1 1 0 0 1 0 0]
```

Subsequently, we analyzed the metrics of it. The, we save the model to use it in the html application.



```
Accuracy Score: 0.9487179487179487
Precision Score: 0.9836065573770492
True positive Rate: 0.9230769230769231
False positive Rate 0.019230769230769232
F1 Score: 0.9523809523809524
Specificity: 0.9807692307692307
Mean Absolute Error: 0.05128205128205128
ROC Area: 0.951923076923077
```

Save and load the model

As the metrics are satisfatory, we can save and load the model.

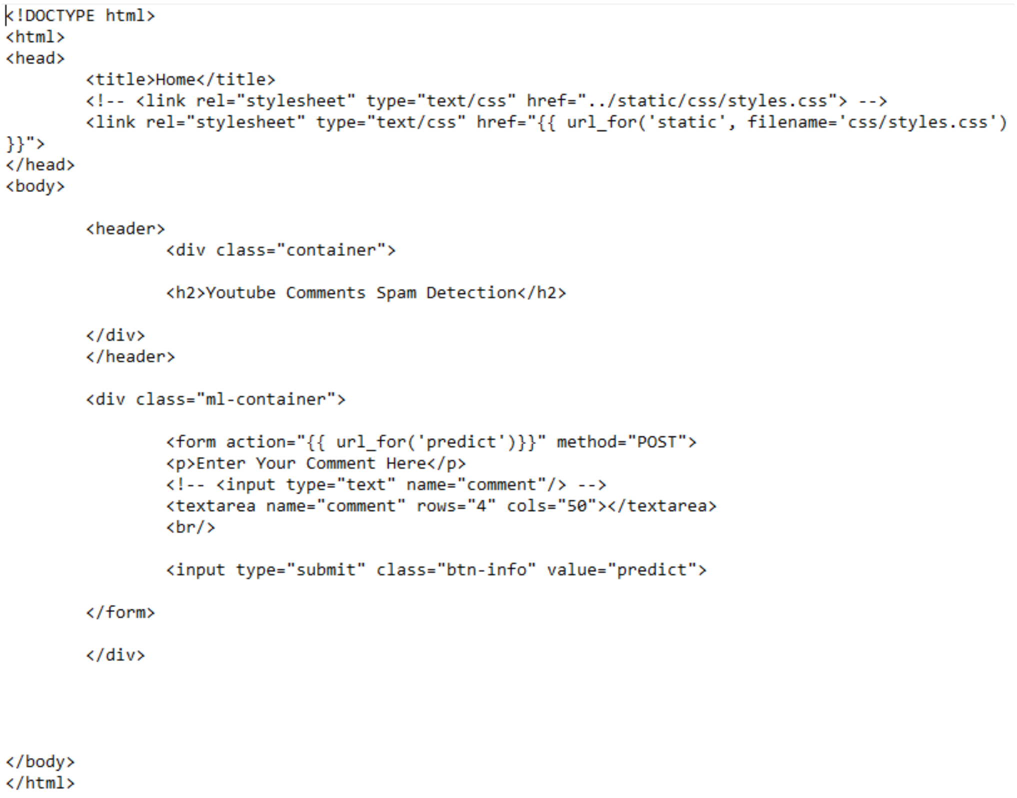
```
In [11]: import pickle              # pickle used for serializing and de-serializing a Python object structure

         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
```

```
In [12]: # Load the model
         ytb_model = open("model.pkl","rb")           # open the file for reading
         new_model = pickle.load(ytb_model)           # load the object from the file into new_model
         new_model

Out[12]: SVC(kernel='linear', random_state=0)
```

Web Application

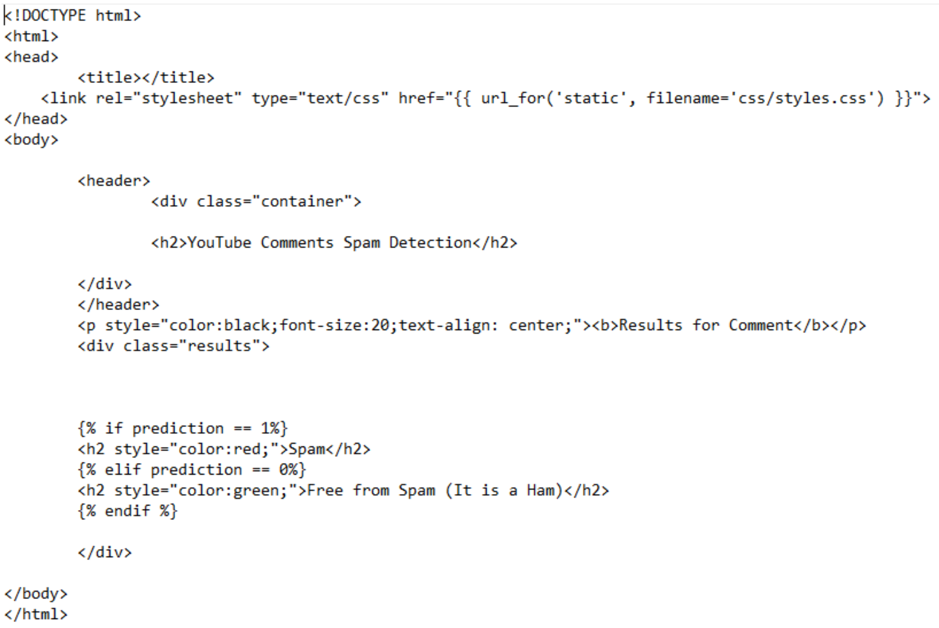
This page consists of a simple web application with a form field so the user can type a message. Then, it will render a classification spam or ham comment. We have two HTML files, *home.html* and *results.html*.

Figure 2. Home.HTML code



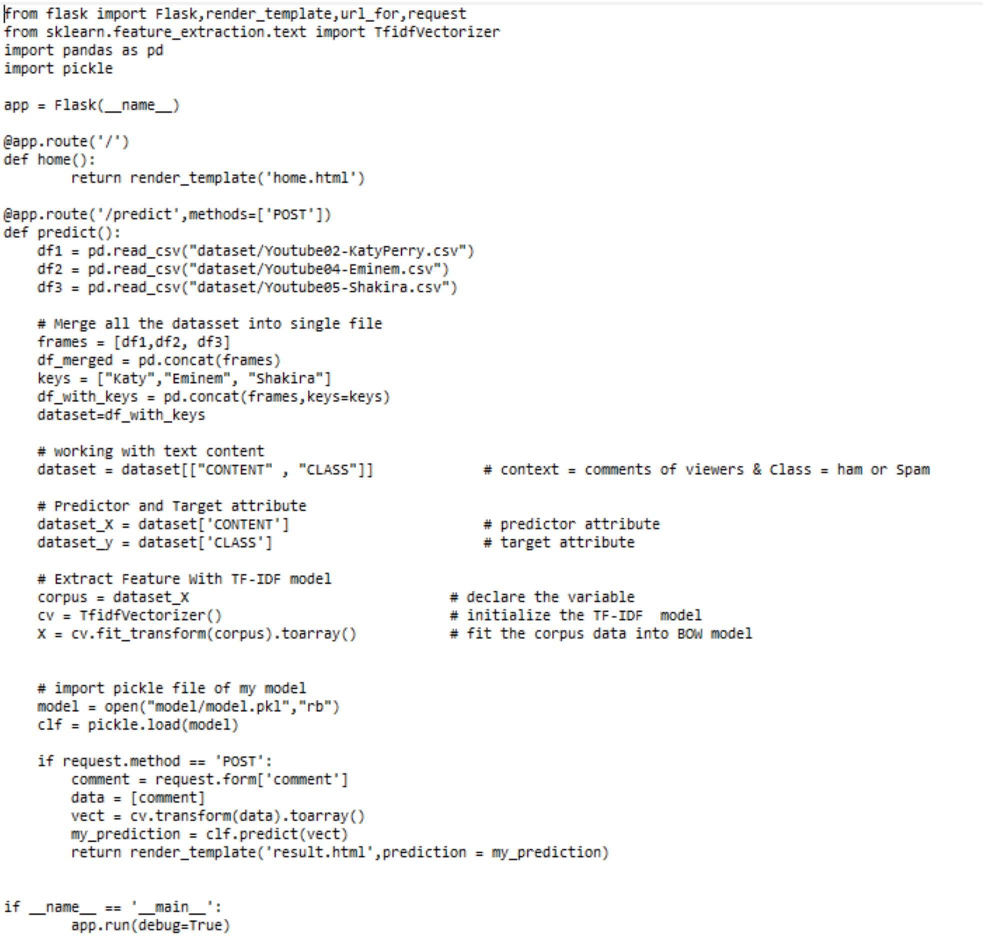
```
<!DOCTYPE html>
<html>
<head>
        <title>Home</title>
        <!-- <link rel="stylesheet" type="text/css" href="../static/css/styles.css"> -->
        <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='css/styles.css')
}}">
</head>
<body>

        <header>
                <div class="container">

                <h2>Youtube Comments Spam Detection</h2>

        </div>
        </header>

        <div class="ml-container">

                <form action="{{ url_for('predict')}}" method="POST">
                <p>Enter Your Comment Here</p>
                <!-- <input type="text" name="comment"/> -->
                <textarea name="comment" rows="4" cols="50"></textarea>
                <br/>

                <input type="submit" class="btn-info" value="predict">

        </form>

        </div>



</body>
</html>
```

Figure 3. Result.HTML code



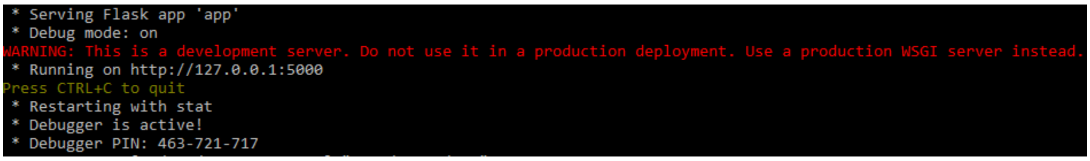
```
<!DOCTYPE html>
<html>
<head>
        <title></title>
    <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='css/styles.css') }}">
</head>
<body>

        <header>
                <div class="container">

                <h2>YouTube Comments Spam Detection</h2>

        </div>
        </header>
        <p style="color:black;font-size:20;text-align: center;"><b>Results for Comment</b></p>
        <div class="results">



        {% if prediction == 1%}
        <h2 style="color:red;">Spam</h2>
        {% elif prediction == 0%}
        <h2 style="color:green;">Free from Spam (It is a Ham)</h2>
        {% endif %}

        </div>

</body>
</html>
```

The *app.py* file contains the main code that will be executed by the Python interpreter to run the flask application, it includes the ML for classifying.



```
from flask import Flask,render_template,url_for,request
from sklearn.feature_extraction.text import TfidfVectorizer
import pandas as pd
import pickle

app = Flask(__name__)

@app.route('/')
def home():
        return render_template('home.html')

@app.route('/predict',methods=['POST'])
def predict():
    df1 = pd.read_csv("dataset/Youtube02-KatyPerry.csv")
    df2 = pd.read_csv("dataset/Youtube04-Eminem.csv")
    df3 = pd.read_csv("dataset/Youtube05-Shakira.csv")

    # Merge all the datasset into single file
    frames = [df1,df2, df3]
    df_merged = pd.concat(frames)
    keys = ["Katy","Eminem", "Shakira"]
    df_with_keys = pd.concat(frames,keys=keys)
    dataset=df_with_keys

    # working with text content
    dataset = dataset[["CONTENT" , "CLASS"]]             # context = comments of viewers & Class = ham or Spam

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

    # 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


    # import pickle file of my model
    model = open("model/model.pkl","rb")
    clf = pickle.load(model)

    if request.method == 'POST':
        comment = request.form['comment']
        data = [comment]
        vect = cv.transform(data).toarray()
        my_prediction = clf.predict(vect)
        return render_template('result.html',prediction = my_prediction)


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

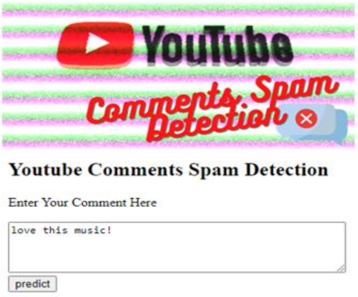
Afterwards, we can run the API on the terminal.



```
 * 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: 463-721-717
```

Scenarios on Web Application

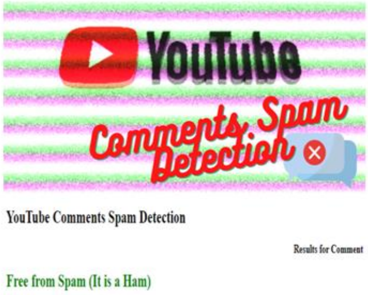
The first scenario is the one there is a ham comment.



Youtube Comments Spam Detection

Enter Your Comment Here

love this music!

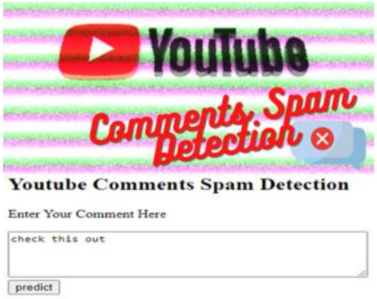


YouTube Comments Spam Detection

Results for Comment

Free from Spam (It is a Ham)

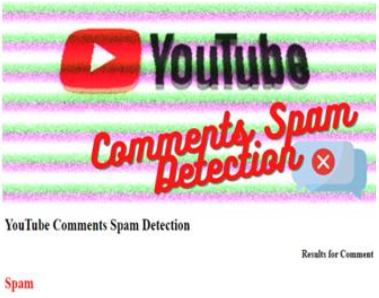
Other possibility is the spam comment.



Youtube Comments Spam Detection

Enter Your Comment Here

check this out



YouTube Comments Spam Detection

Results for Comment

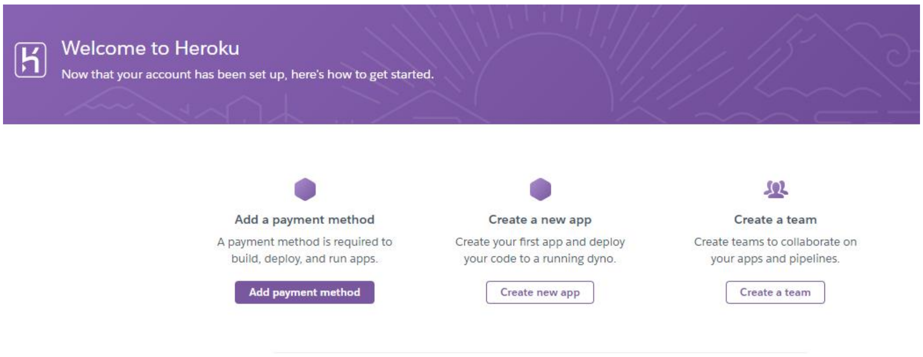
Spam

So, it works just fine.

Model deployment using Heroku

Thereafter that our model has been trained, the machine learning pipeline has been set up, and the application has been tested locally, we may start the Heroku deployment. We linked our GitHub repository to the Heroku account.

1) Heroku – creating a new app



Welcome to Heroku

Now that your account has been set up, here's how to get started.

Add a payment method

A payment method is required to build, deploy, and run apps.

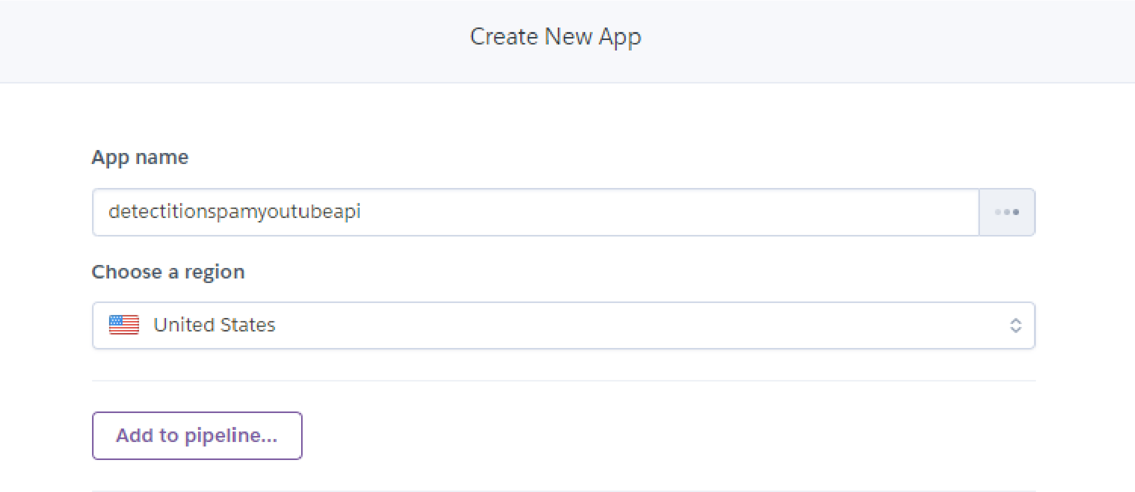
Create a new app

Create your first app and deploy your code to a running dyno.

Create a team

Create teams to collaborate on your apps and pipelines.

2) Enter App name and region



Create New App

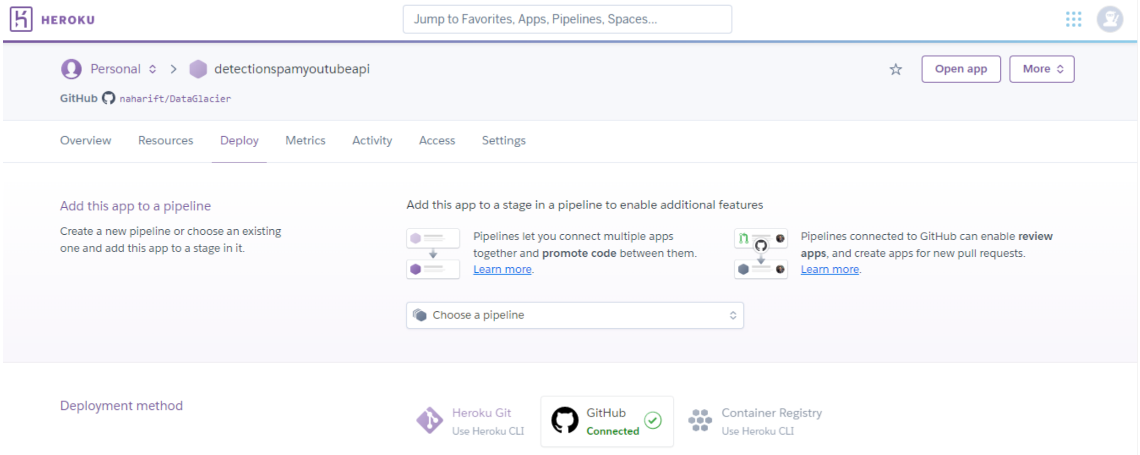
App name

detectitionspamyoutubeapi

Choose a region

United States

3) Connect to GitHub repository.



Add this app to a pipeline

Create a new pipeline or choose an existing one and add this app to a stage in it.

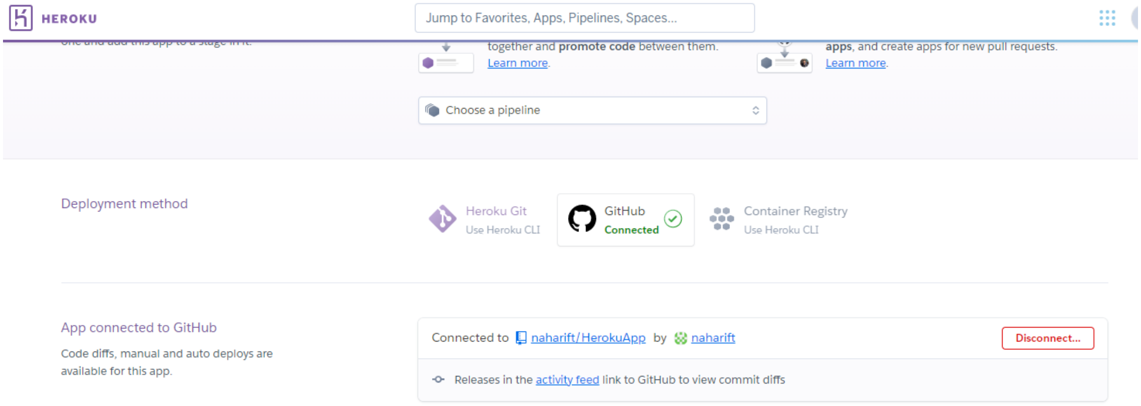
Add this app to a stage in a pipeline to enable additional features

Pipelines let you connect multiple apps together and **promote code** between them. Learn more.

Pipelines connected to GitHub can enable **review apps**, and create apps for new pull requests. Learn more.

Deployment method

4) Then, I choose the repository where the code is.



together and **promote code** between them. Learn more.

apps, and create apps for new pull requests. Learn more.

Deployment method

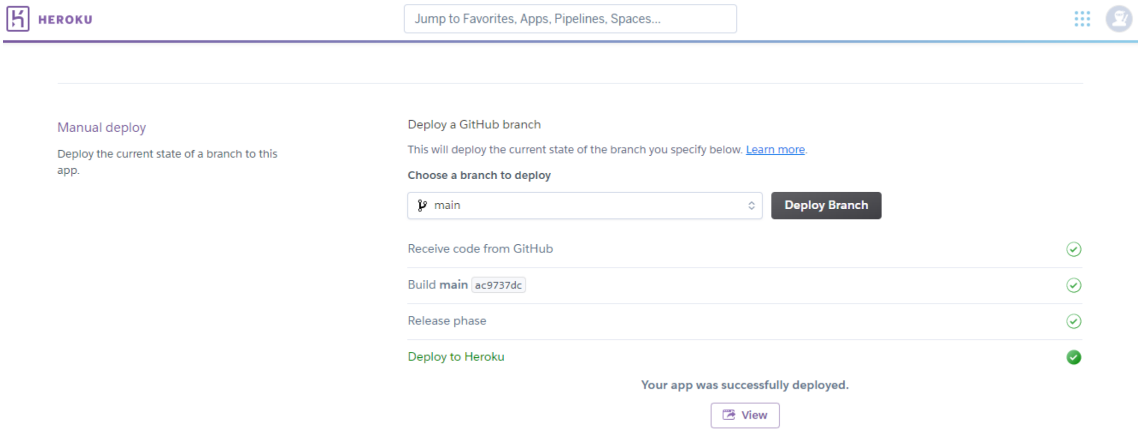
App connected to GitHub

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

Connected to naharift/HerokuApp by naharift

Releases in the activity feed link to GitHub to view commit diffs

5) Deploy branch.



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

Receive code from GitHub ✓

Build **main** ac9737dc ✓

Release phase ✓

Deploy to Heroku ✓

Your app was successfully deployed.

We must wait some minutes to get our application ready.

It’s possible to check on: https://detectionspamyoutubeapi.herokuapp.com/