Data Glacier

Data Science Internship at Data Glacier

Week 4: Deployment on Flask

Name: Amrapali Mhaisgawali

Batch code: LISUM14

Submission date: 26/10/2022

Submitted to: Data Glacier

Index:

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.

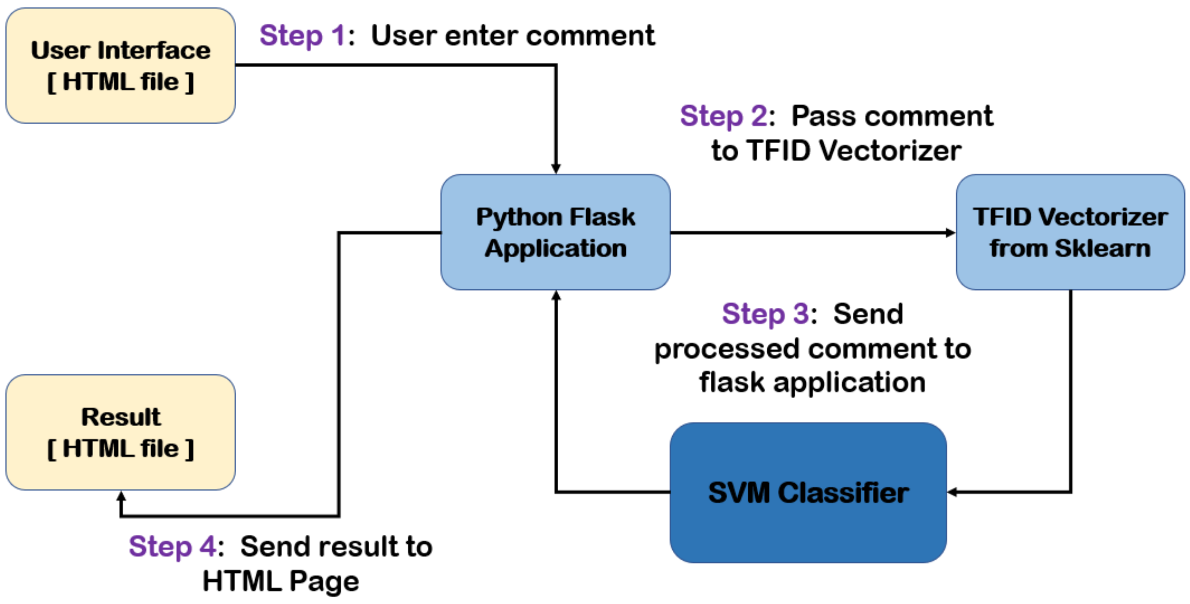

User Interface [HTML file]
Step 1: User enter comment
Step 2: Pass comment to TFID Vectorizer
Python Flask Application
TFID Vectorizer from Sklearn
Step 3: Send processed comment to flask application
Result [HTML file]
SVM Classifier
Step 4: Send result to HTML Page

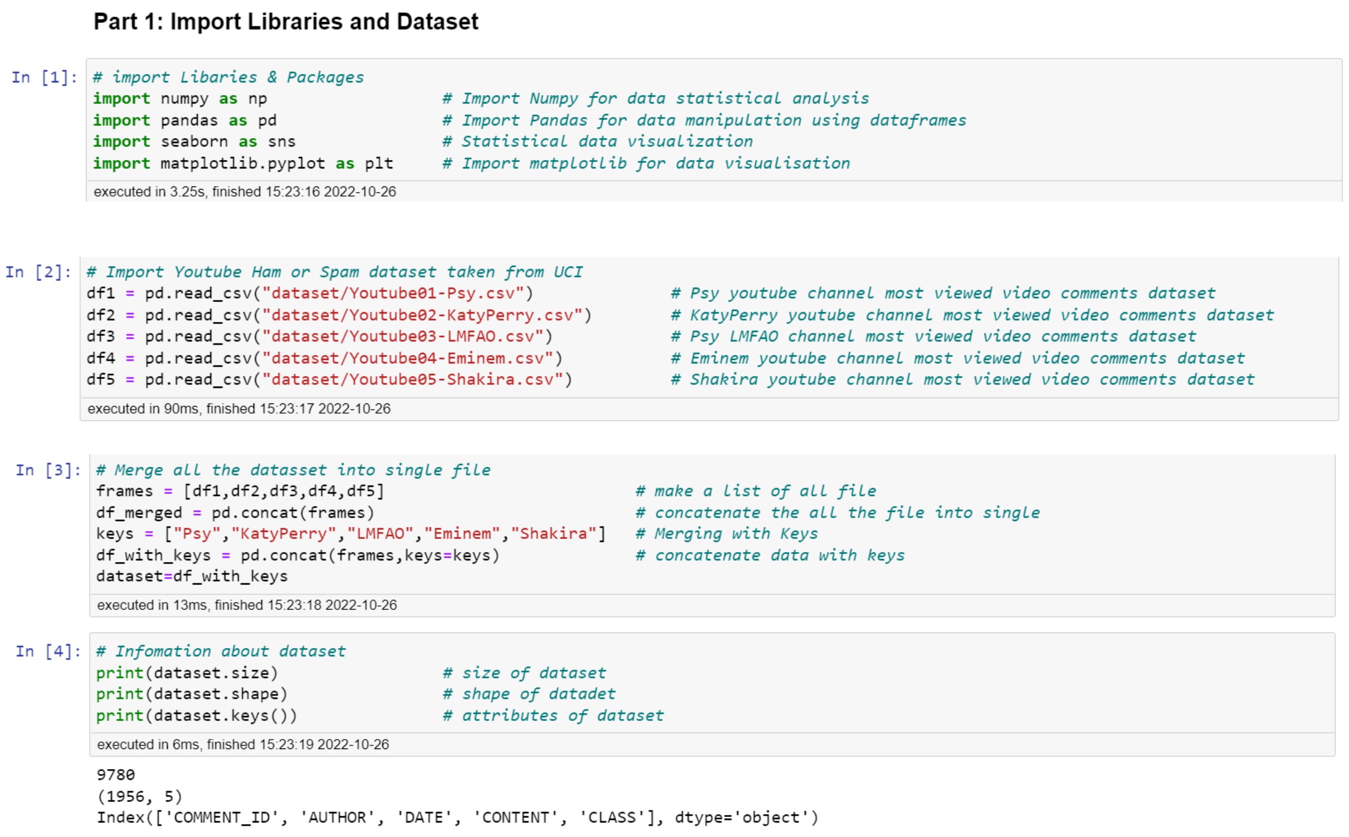
Fig. 2.1 Application Workflow

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.

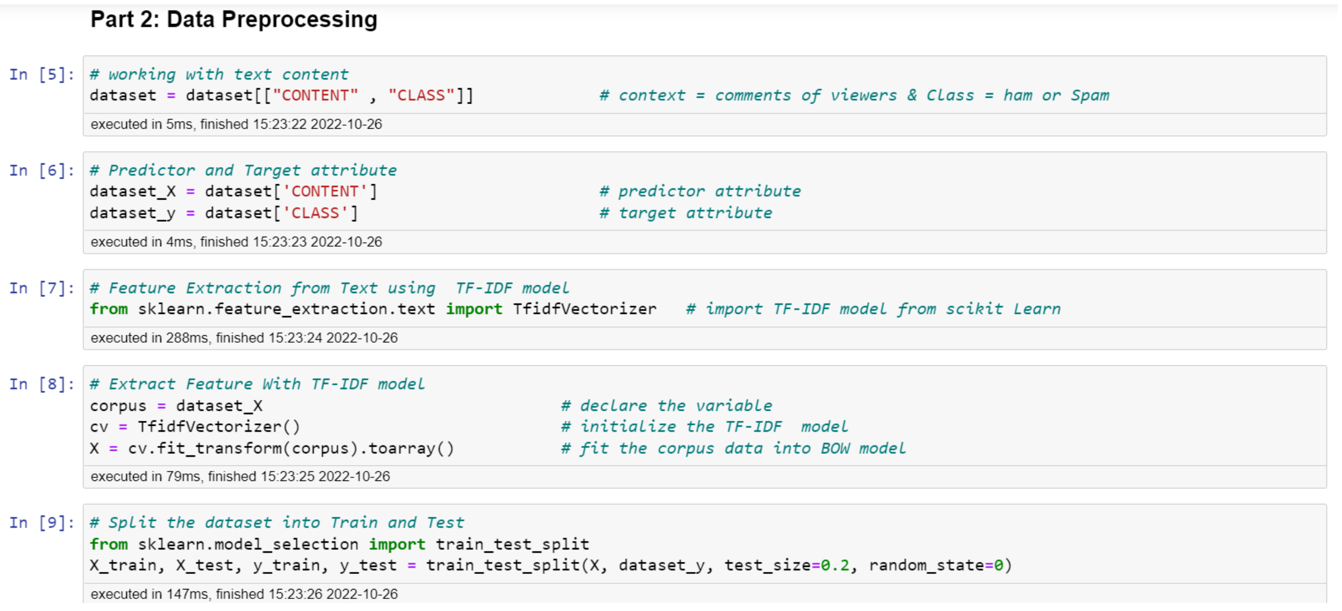


Part 1: Import Libraries and Dataset

```
In [1]: # import Libaries & 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 datasset 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]: # Infomation about dataset
        print(dataset.size)                 # size of dataset
        print(dataset.shape)                # shape of datadet
        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.

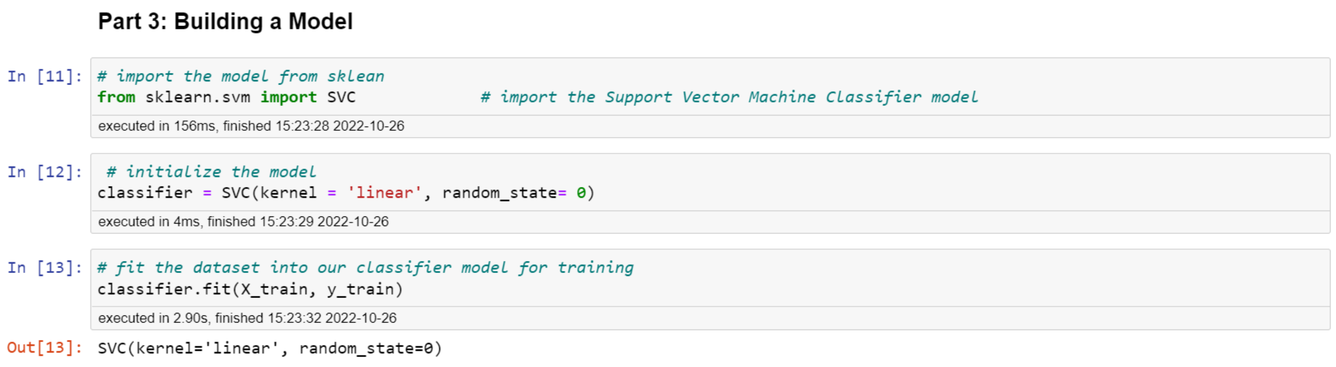


Part 2: Data Preprocessing

```
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.

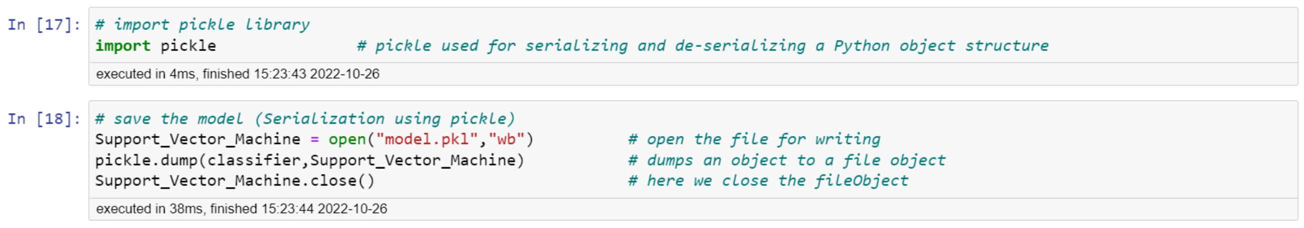


Part 3: Building a Model

```
In [11]: # import the model from sklean
         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.

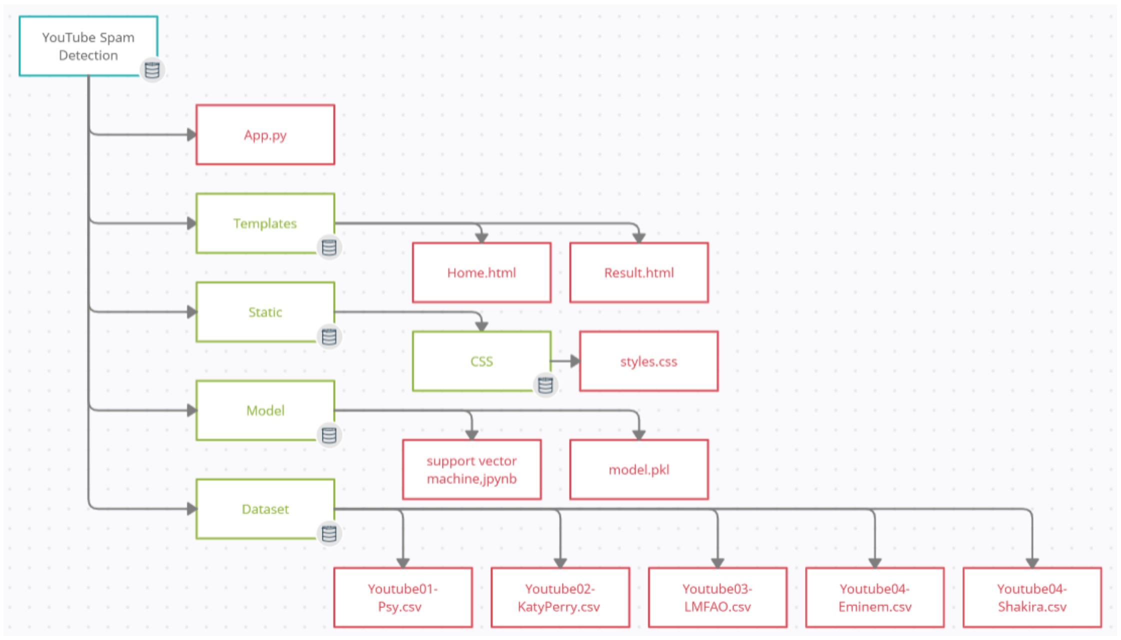
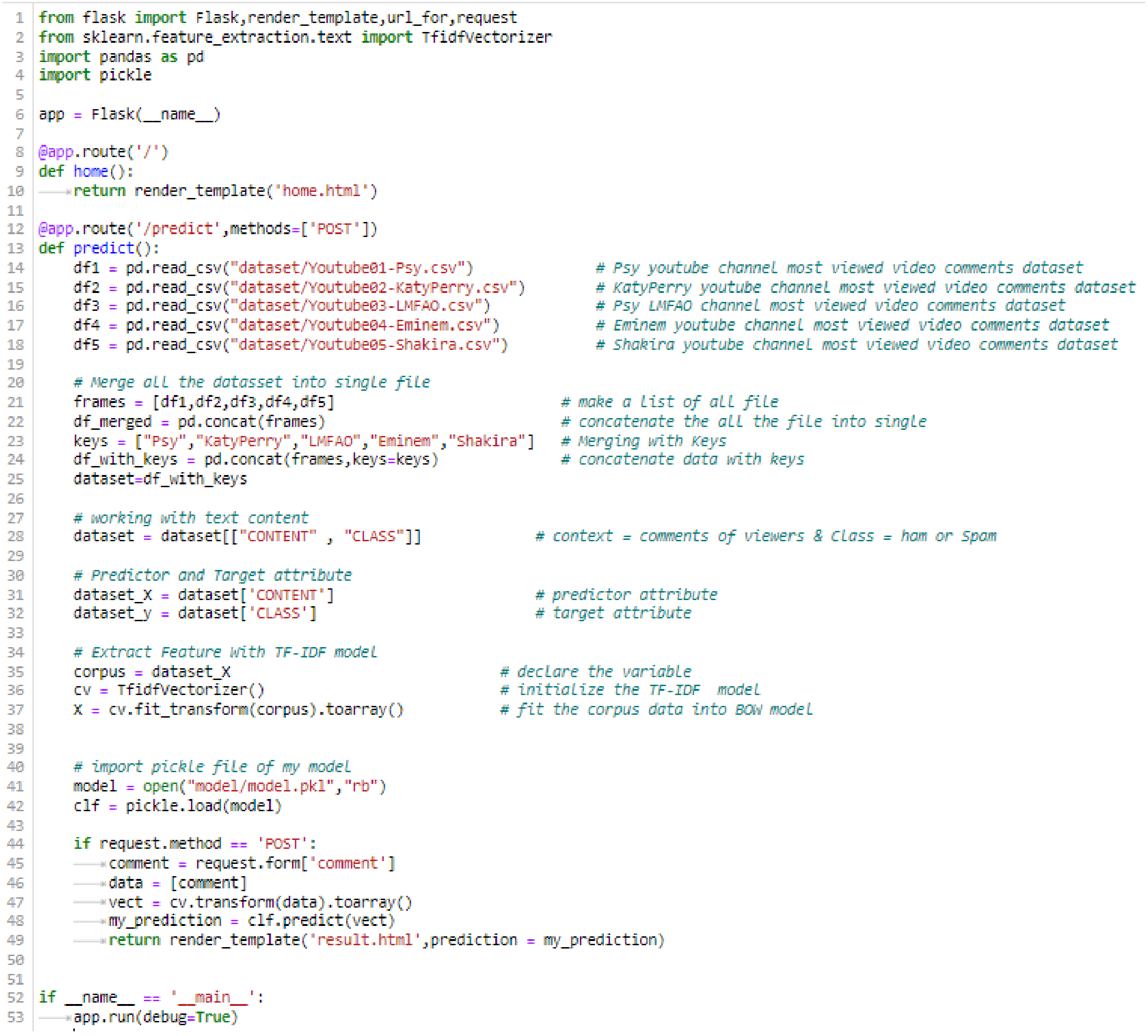


Fig 4.1 Directory Structure

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.

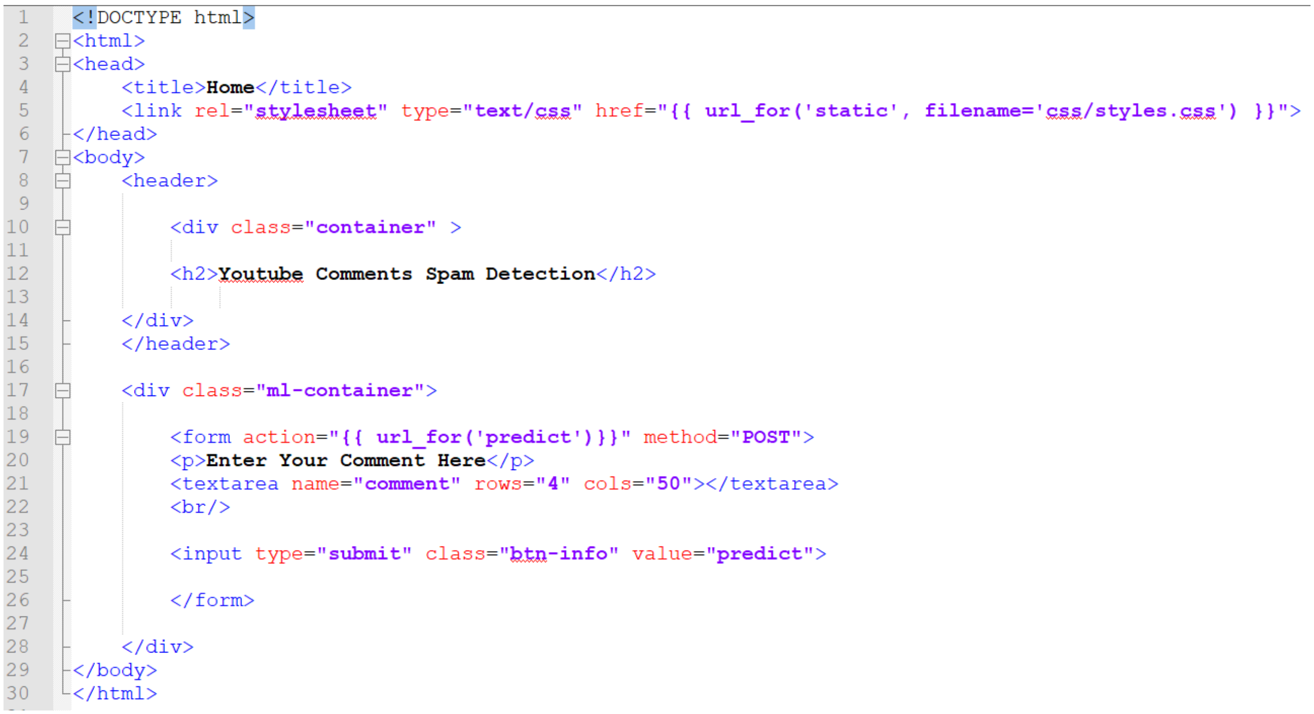


```python
from flask import Flask,render_template,url_for,request
from sklearn.feature_extraction.text import TfidfVectorizer
import pandas as pd
import pickle

app = Flask(__name__)

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

@app.route('/predict',methods=['POST'])
def predict():
    df1 = pd.read_csv("dataset/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

    # 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

    # 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)
```

- 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

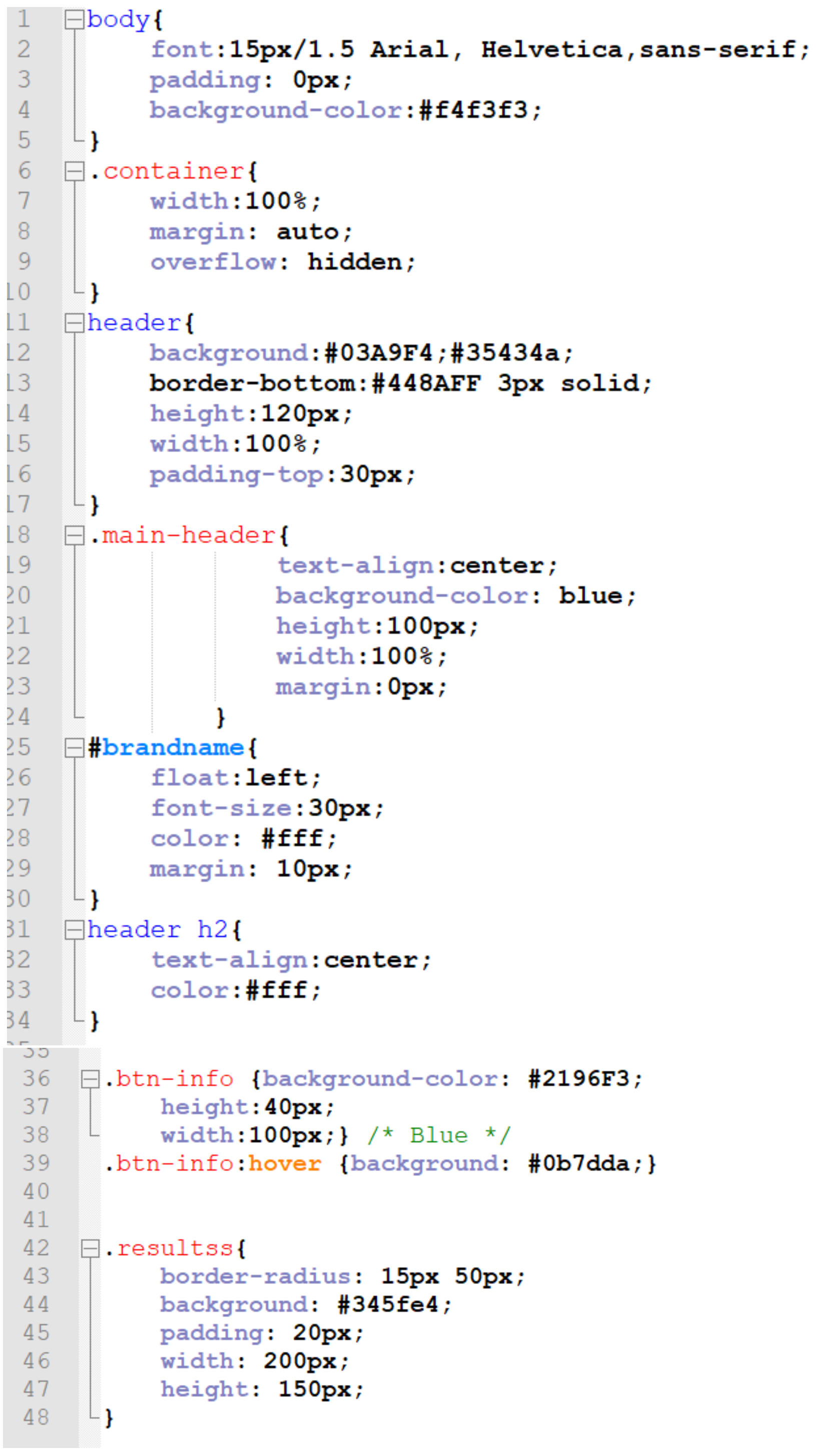
The home.html file will render a text form where a user can enter a comment.



```
<!DOCTYPE html>
<html>
<head>
    <title>Home</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>

    <div class="ml-container">

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

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

        </form>

    </div>
</body>
</html>
```

4.2 Styles.css

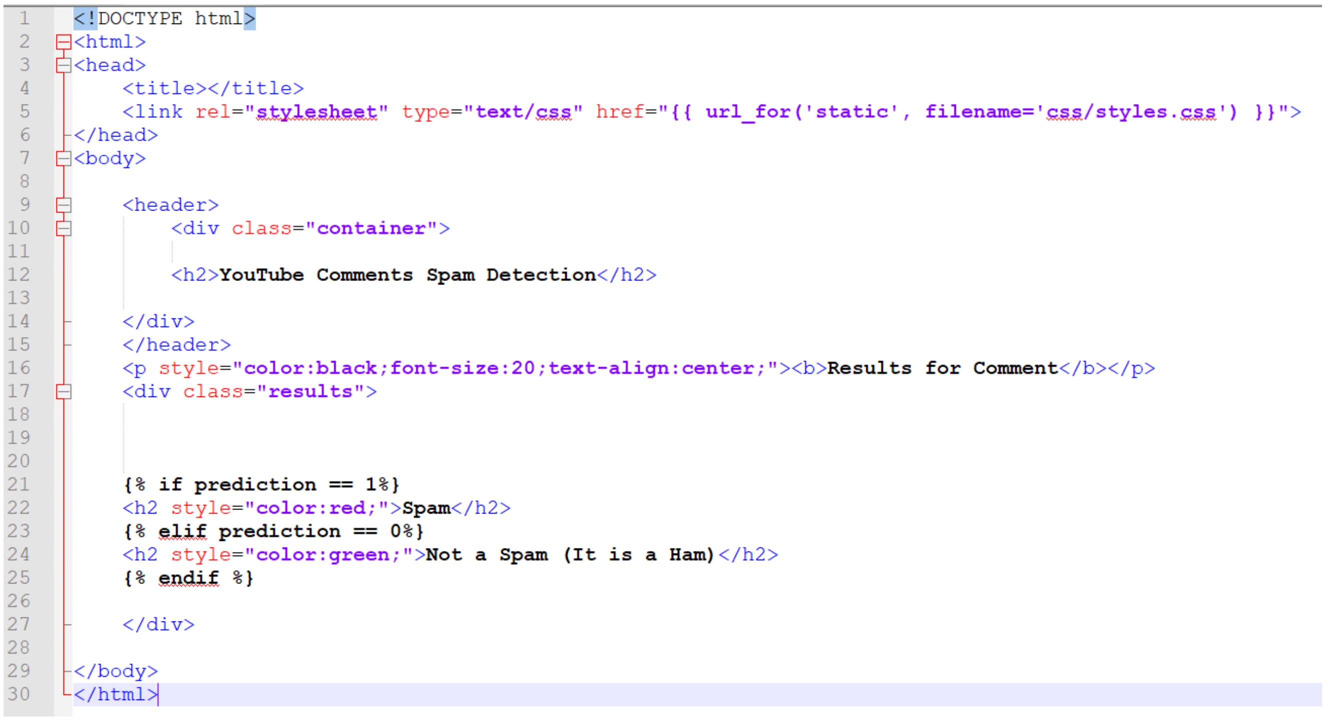
CSS is to determine how the look and feel of HTML documents. styles.css must be saved in a sub-directory called static, which is the default directory where Flask looks for static files such as CSS.



```css
body{
    font:15px/1.5 Arial, Helvetica,sans-serif;
    padding: 0px;
    background-color:#f4f3f3;
}
.container{
    width:100%;
    margin: auto;
    overflow: hidden;
}
header{
    background:#03A9F4;#35434a;
    border-bottom:#448AFF 3px solid;
    height:120px;
    width:100%;
    padding-top:30px;
}
.main-header{
            text-align:center;
            background-color: blue;
            height:100px;
            width:100%;
            margin:0px;
        }
#brandname{
    float:left;
    font-size:30px;
    color: #fff;
    margin: 10px;
}
header h2{
    text-align:center;
    color:#fff;
}

.btn-info {background-color: #2196F3;
    height:40px;
    width:100px;} /* Blue */
.btn-info:hover {background: #0b7dda;}


.resultss{
    border-radius: 15px 50px;
    background: #345fe4;
    padding: 20px;
    width: 200px;
    height: 150px;
}
```

4.4 Result.html

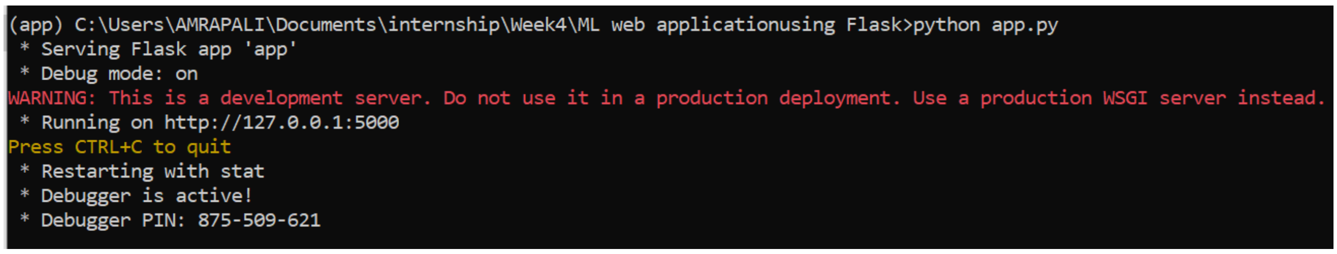
Result.html file will be rendered via the render_template('result.html', prediction=my_prediction) which is inside the predict function of app.py script to display the text that a user-submitted via the text field. Result.html has Jinja syntax : {% if prediction ==1%},{% elif prediction == 0%},{% endif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



```html
<!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;">Not a Spam (It is a Ham)</h2>
    {% endif %}

    </div>

</body>
</html>
```

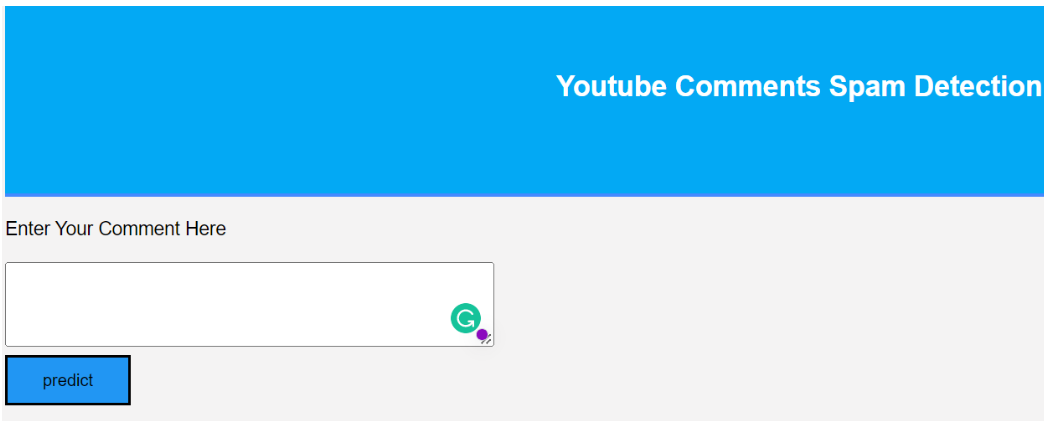
4.5 Results with running procedure

Following is a command to run app.py

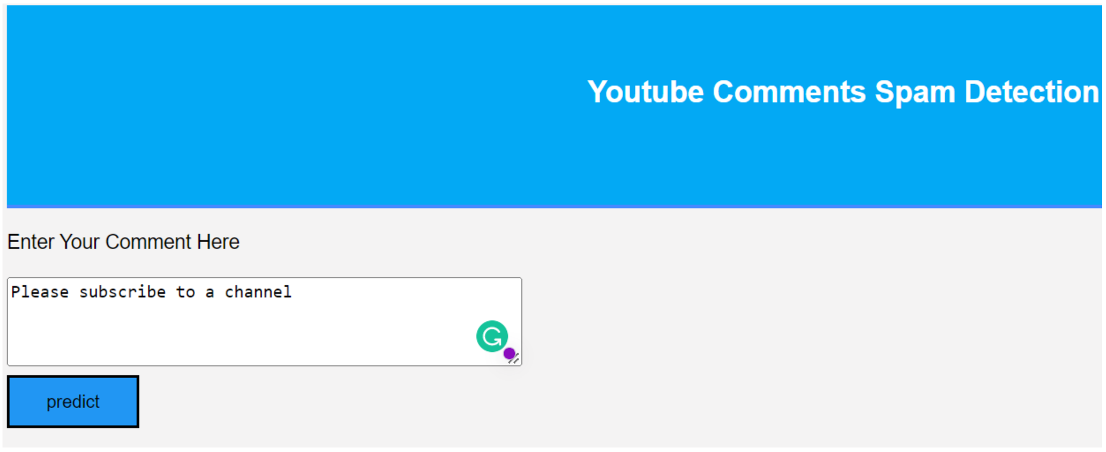


```
(app) C:\Users\AMRAPALI\Documents\internship\Week4\ML web applicationusing Flask>python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press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

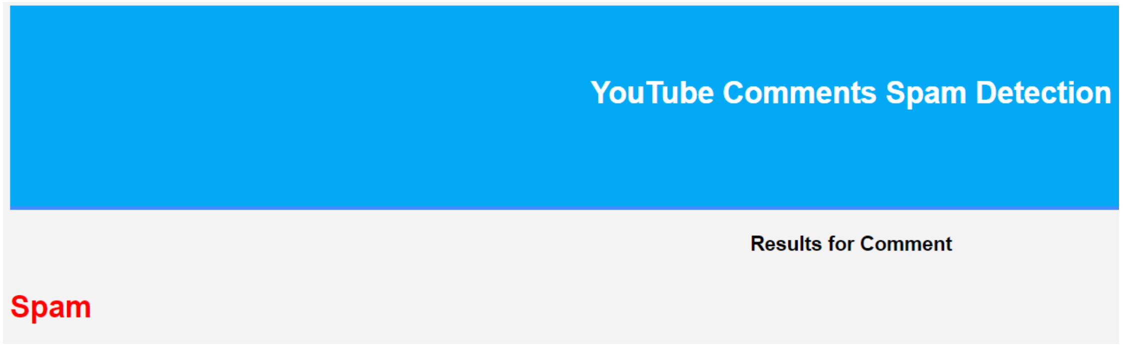
Now open a web browser and navigate to http://127.0.0.1:5000/ following is output of home.html.



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



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