Data Glacier: Data Science Intern

Week 5: Cloud API Development

Name: Ratna Manjeera Grandhi
Batch Code: LISUM23
Date: August 4th, 2023
Submitted to: Data Glacier

Table Of Contents

1. Introduction

In this project, I am going to be designing and deploying a machine learning model (SVM) using the Flask Methodology. As a simple example demonstration for our model, it would help a comment to be screened to determine if it's a spam comment or a genuine comment.

I will be focusing on building a machine learning model for YouTube Comments, then create an API for that specific model, using Flask language, Python micro-framework for constructing web applications. This framework will allow us to utilize predictive capabilities through HTTP requests.

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.

Table 2.1: Dataset Information

Dataset	**YouTube ID**	**Spam**	**Legit**	**Total**
Shakira	89dafert34rer	150	150	300
KatyPerry	afAFDS234ASF	125	125	250
J Lo	DFqrwqwt23asf	110	207	317
Eminem	ertYI234Uer	434	567	1001

2.2 Attribute Information

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

Table 2.2: Attribute Information

Attributes	Example
Comment Type	LAiadnofaAFfgAFSEDUHIKB
Commenter	Jason RY
Date	2023-04-01 T 11:20:45
Text	Awesome video checkout this channel

Class	1(Spam)

3. Building a Model

3.1 Import Required Libraries and Dataset

In this section, we import libraries and dataset which contain the information of five most commented video.

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
```

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
```

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
```

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

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

3.2 Data Preprocessing

The dataset used here is split into 80% for the training set and the remaining 20% for the test set. We fed our dataset into a Term Frequency-Inverse document frequency (TF-IDF) vectorizer which transforms words into numerical features (numpy arrays) for training and testing.

3.3 Build Model

After data preprocessing, we implement machine learning model to classify the YouTube spam comments. For this purpose, we implement Support Vector Machine (SVM) using scikit-learn. After importing and initialize SVM model we fit into training dataset.

3.4 Save the Model

After that we save our model using pickle.

4. Turning Model into Web Application

We develop a web application that consists of a simple web page with a form field that lets us enter a message. After submitting the message to the web application, it will render it on a new page which gives us a result of spam or ham(not spam). First, we create a folder for this project called YouTube Spam Filtering, this is the directory tree inside the folder. We will explain each file.

The sub-directory templates are the directory in which Flask will look for static HTML files for rendering in the web browser, in our case, we have two HTML files: home.html and result.html

4.1 App.py

The app.py file contains the main code that will be executed by the Python interpreter to run the Flask web application, it included the ML code for classifying SD.

• We ran our application as a single module; thus we initialized a new Flask instance 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, we used the route decorator (@app.route('/')) to specify the URL that should trigger the execution of the home function.

• Our home function simply rendered the home.html HTML file, which is located in the templates folder.

• Inside the predict function, we access the spam data set, pre-process the text, and make predictions, then store the model. We access the new message entered by the user and use our model to make a prediction for its label.

• We used the POST method to transport the form data to the server in the message body. Finally, by setting the debug=True argument inside the app.run method, we further activated Flask's debugger.

• Lastly, we used the run function 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__'.

4.2 Home.html

4.3 Style.css

4.4 Results.html

4.5 Running

4.6 Results Sample: