EasyChair Preprint

№ 15711

YouTube Spam Comment Detection System

Siladitya Mukherjee, Soumya Dey and Anal Acharya

January 13, 2025

YouTube Spam Comment Detection System

Prof. Siladitya Mukherjee[1] Soumya Dey[2] and Anal Acharya[3]

[1] Assistant Professor, Department of Computer Science, St. Xavier's College Kolkata, India, siladitya.mukherjee@sxccal.edu
[2] Student, Department of Computer Science, St. Xavier's College Kolkata, India, soumyadey200@hotmail.com
[3] Associste Professor, Department of Computer Science, St. Xavier's College Kolkata, India, anal.acharya@sxccal.edu

Abstract. Spamming involves posting irrelevant and unsolicited comments on social media platforms or video-sharing sites. Bots are used to post these messages, aiming to lower the ranking of the content or disrupt users' viewing experience, which ultimately reduces the rank of the video or post. Besides that, spam comments can also include malicious links that can steal user's sensitive data when clicked. Spamming, often manual, is prevalent on competitive platforms. While data mining can mitigate some forms of spam, this project automates spam comment detection on YouTube using machine learning techniques. We'll leverage a dataset of YouTube spam comments from the UCI Machine Learning Repository and apply the Count-Vectorizer and Support Vector Machine algorithm for clustering on the given dataset using python programming.

Keywords: Spam Detection, Spam Filtering, Supervised Machine Learning, Text Analysis.

1 Literature Survey

Seungwoo Choi et al. proposed and experimented on Ted Talks to check whether the comments are covering the information about the actual video content. But this method alone was found inadequate for analyzing the feelings and the opinions expressed in the comment section of platforms such as YouTube.

YouTube also struggles with malicious users who post low-quality videos, often referred to as video spam. Researchers have explored techniques to extract meaningful features from metadata like view counts, titles, and descriptions to identify such content.

Alex Kantchelian et al.[1] proposed a Spam Detection (SD) method that leverages text informativeness. They suggested expanding the definition of spam to include URLs and short messages, and proposed further advancements like incorporating antagonist awareness and real-time online prediction capabilities.

Typically, spam detection focuses on textual content. The Tangram framework collects vast amounts of data from social networks to identify and generate spam templates. These templates serve as a benchmark to rapidly classify incoming messages as spam or non-spam, significantly reducing processing time.

Enhua Tan et al.[2] developed a runtime spam detection system called BARS. This system maintains a database of known spam URLs and compares each new post's URL against this blacklist. Additionally, clustering user IDs based on shared URLs enhances detection accuracy. However, the effectiveness of BARS is heavily reliant on the quality and maintenance of the blacklisted URL database.

M. McCord et al. developed a machine learning algorithm where training with content and user-centered facets were necessary to detect spam comments and the identity of the user who is making these comments. Random Forest Classifier had been found to be the best one, offering the correct results.

The authors, Qingxi Peng et al. operated another different technique to determine the spam comments which is known as Sentiment analysis. They used to conclude their results by suggesting improvements to the calculated sentiment score that was sordid of the paper.

Igor Santos and colleagues employed anomaly detection to classify emails as spam or legitimate, leveraging the deviation from normal email patterns. This approach proved effective, especially in scenarios with limited labeled training data, as often encountered in traditional classification systems.

While blocking spammers is a common technique, it's less effective on YouTube, where spam is frequently posted by real users for self-promotion on popular videos. This makes detection more challenging due to the similarity to genuine comments.

As highlighted by Bratko et al.,[4] spam filtering differs from standard text categorization tasks. Cross-validation isn't optimal, as earlier data should be used for training and later data for testing. Moreover, misclassifications have unequal consequences: blocking a legitimate message is more severe than allowing spam.

This paper presents an online spam detection system capable of real-time inspection of user-generated messages. The system can be seamlessly integrated into any online social network platform.

2 Proposed System Architecture

In this project, Count Vectorization is used to extract features from the dataset, which is then split into training and testing sets. A Term Frequency-Inverse Document Frequency (TF-IDF) vectorizer converts the text data into numerical features suitable for machine learning models. These numerical features are then used to train and make

predictions with Support Vector Machines (SVM), Logistic Regression (LR), and Multi-Layer Perceptron (MLP) models.

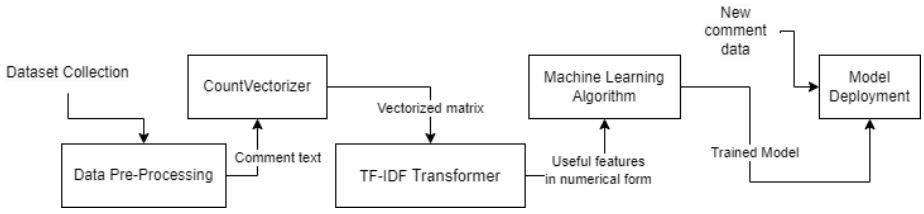

Fig. 1. Steps in Proposed Architecture.

3 Introduction

YouTube, the world's largest video-sharing platform today, was founded in 2005 and subsequently acquired by Google in 2006.

Youtube is the home to content creators, influencers, and most importantly the media, supporting every genre of videos starting from entertainment to education to politics and many others. People who frequently upload videos on Youtube are called YouTubers and they can earn money through Youtube if they get monetized by clearing some threshold.

But, as the saying goes “it’s a boon and a bane”, Youtube has its disadvantages. Youtube allows the audience or the viewers to interact with the YouTubers with a feature called a Comment Section. One can appreciate the video, ask questions or give constructive criticism. But, human nature is not that simple and some viewers just use the comments feature to abuse the uploader or post something harmful like malicious and inappropriate links or promote their products on Youtube channels. These are considered spam comments as these comments are not helpful. Youtube has its own spam filter but even then some of the comments get through.

In this paper, we are trying to detect the percentages of spam comments in a particular youtube video using Supervised Machine Learning algorithms coded in Python language. We use a Countvectorizer on the Youtube SpamComments Dataset obtained from UCI Machine Learning Repository and apply various Supervised learning algorithms and make a comparative study, as to which algorithm generates the highest accuracy. The system can be deployed as a component of any online social network platform.

4 Aim and Objectives

4.1 Purpose of the project

- To conduct a literature survey on spam comment detection systems.
- To arrive at the requirement specification for a spam comment detection system using the YouTube dataset.
- To design and develop a spam comment detection model for Youtube videos.

- To implement and compare contemporary machine learning techniques for text analysis.
- To document the report by unifying all the results and outcomes.

5 Scope of the project

5.1 In Scope

- A spam comment detection model that can detect spam comments.
- A system that can filter out unnecessary information from the comment text to improve accuracy.

5.2 Out of Scope

- Process comments in any language other than English.

6 Preparing The Data

6.1 Data Collection

YouTube Spam Collection. The YouTube Spam Collection dataset is a publicly available resource used for spam research. It contains 1,956 genuine comments taken from five of the top 10 most-viewed YouTube videos during a particular time frame.

The collection is composed of one CSV file per dataset, with the following attributes: COMMENT_ID, AUTHOR, DATE, CONTENT, CLASS.

The table below shows the details of each dataset, including its corresponding YouTube video ID, the count of spam and non-spam comments, and the total number of comments.

Table 1. No. of samples in each class in each dataset.

Dataset	Spam	Not Spam	Total
Psy	175	175	350
KatyPerry	175	175	350
LMFAO	236	202	438
Eminem	245	203	448
Shakira	174	196	370
Total	1005	951	1956

SMS Spam Collection. The SMS Spam Collection is a freely available resource for mobile phone spam research. It combines several parts:

Manually labeled spam messages: 425 SMS messages identified as spam were hand-picked from the Grumbletext website (http://www.grumbletext.co.uk/).

Randomly selected legitimate messages: A subset of 3,375 SMS messages was chosen at random from the NUS SMS Corpus (http://www.comp.nus.edu.sg/~rpnlpir/downloads/corpora/smsCorpus/). This corpus, containing roughly 10,000 messages, was originally collected by the National University of Singapore for research purposes.

Additional legitimate messages: 450 SMS messages categorized as legitimate were included from Caroline Tag's PhD thesis, titled "A Corpus Linguistics Study of SMS Text Messaging" (available at http://etheses.bham.ac.uk/253/1/Tagg09PhD.pdf).

6.2 **Data Pre-Processing**

We will read the datasets one by one.

- Concatenate the datasets into a single data frame.
- Drop the following columns from the data frame.
    - COMMENT_ID, AUTHOR, DATE
- Apply a function to remove the HTML tags from the values of the "CONTENT" column.[1]
- Apply a function to remove any punctuations, symbols, numbers, and special characters from the value of the "CONTENT" column.[1]
- Split the whole data into train and test sets. (80/20 ratio)
- Use CountVectorizer on the training set for text preprocessing, tokenizing, and filtering of stopwords. This process constructs a dictionary of features and converts the documents into feature vectors.
- Calculate "Term Frequency" (TF) and "Term Frequency multiplied by Inverse Document Frequency" (TF-IDF).
    - Compute Term Frequency (TF): Term Frequency is a measure of how frequently a term appears in a document. It is calculated as:

$$TF(t,d) = \frac{Number\ of\ times\ term\ t\ appears\ in\ document\ d}{Total\ number\ of\ terms\ in\ document\ d}$$

    - Compute Inverse Document Frequency (IDF): Inverse Document Frequency is a measure of how important a term is in the entire dataset. It is calculated as:

$$IDF(t) = log\left(\frac{Total\ number\ of\ documents}{Number\ of\ documents\ containing\ term\ t}\right)$$

- Compute TF-IDF: TF-IDF is a combination of Term Frequency and Inverse Document Frequency. It is calculated as:

$$TF\ \ IDF(t,d)\ =\ TF(t,d) \times IDF(t)$$

- Fit and transform the train set with the “tf-idf” transformer.

6.3 **Training and testing the model**

In all the following tables Class 0 indicates “Not Spam” and Class 1 indicates “Spam”.

Logistic Regression. It is most suitable for binary classification.

All default values. model = LogisticRegression(), Test Accuracy = 80.60%

Table 2. Classification Report for Logistic Regression with all default values as parameters.

Class	Precision	Recall	F1-score	Support
0	0.81	0.81	0.81	12541
1	0.80	0.80	0.80	11832

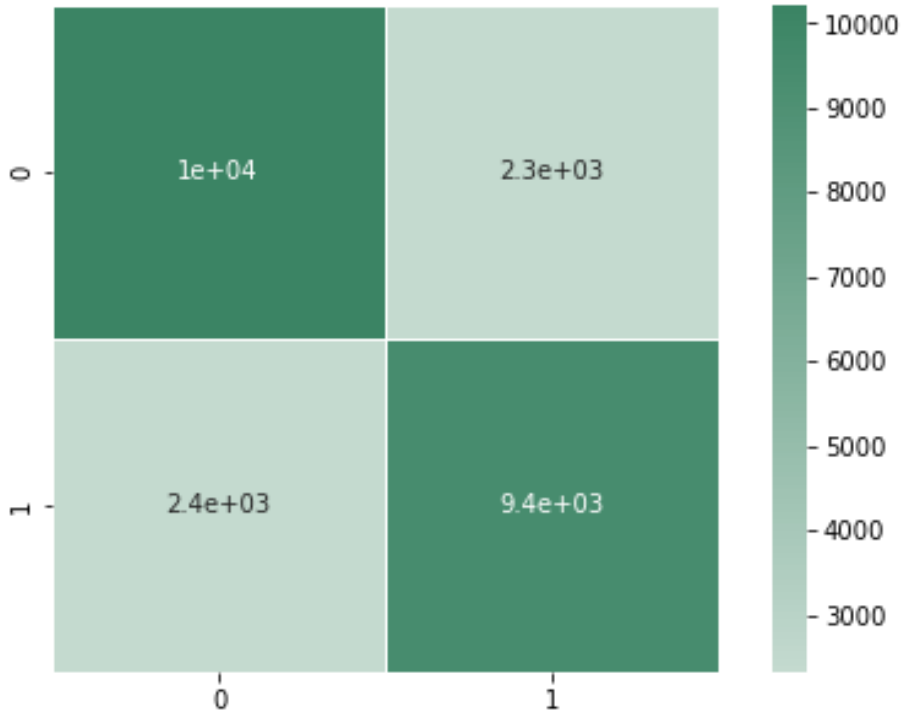

Fig. 2. Confusion matrix for Logistic Regression with all default values as parameters.

C = 2, max_iter = 300. model = LogisticRegression(C = 2.0, max_iter = 300), Test Accuracy = 81.84%

Table 3. Classification Report for Logistic Regression with custom parameters.

Class	Precision	Recall	F1-score	Support
0	0.83	0.82	0.82	12541
1	0.81	0.82	0.81	11832

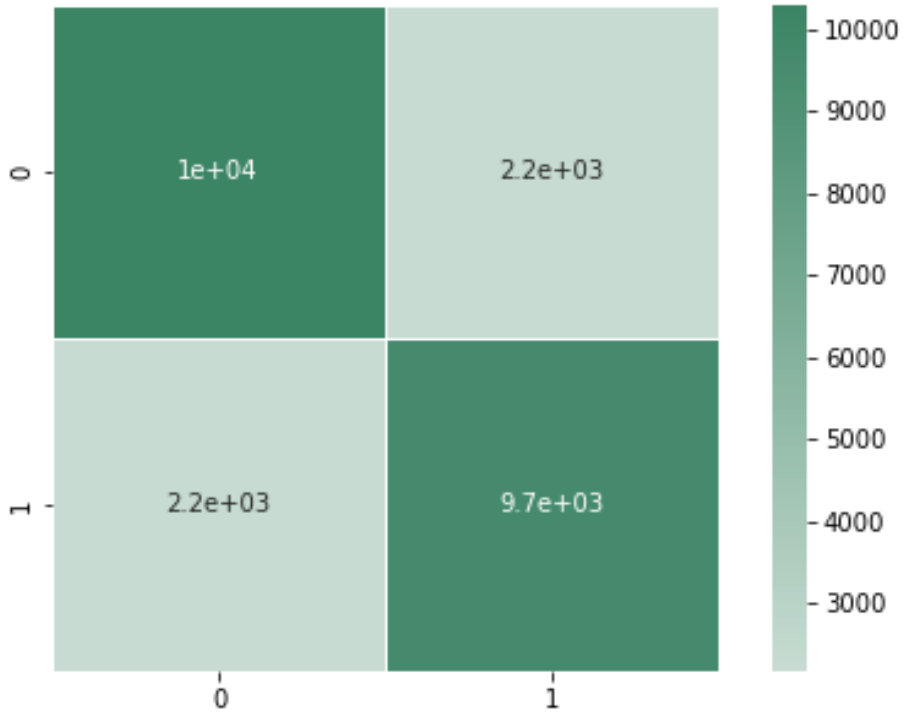

Fig. 3. Confusion matrix for Logistic Regression with custom parameters.

Support Vector Machine (SVM). SVM represents the examples as points in space, and categories are divided by a clear gap that is as wide as possible.

All default values. model = LinearSVC(), Test Accuracy = 83.34%

Table 4. Classification Report for Support Vector Machine with all default values as parameters.

Class	Precision	Recall	F1-score	Support

8

0	0.84	0.83	0.84	12541
1	0.82	0.84	0.83	11832

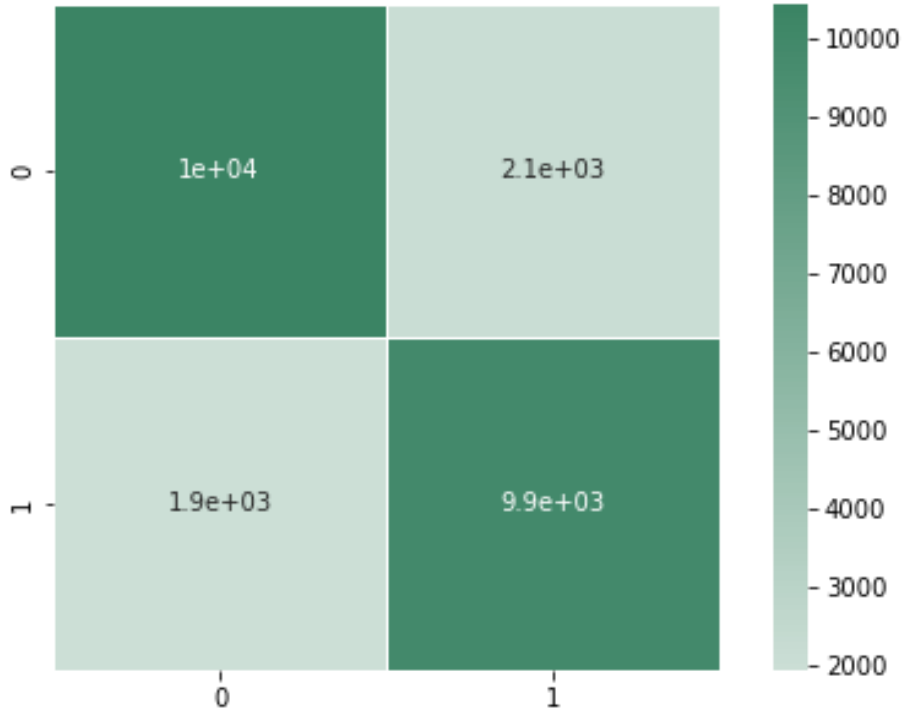

Fig. 4. Confusion matrix for Support Vector Machine with all default values as parameters.

kernel = 'linear'. model = SVC(kernel = 'linear'), Test Accuracy = 89.41%

Table 5. Classification Report for Support Vector Machine with linear kernel.

Class	Precision	Recall	F1-score	Support
0	0.85	0.95	0.90	1142
1	0.94	0.84	0.89	1163

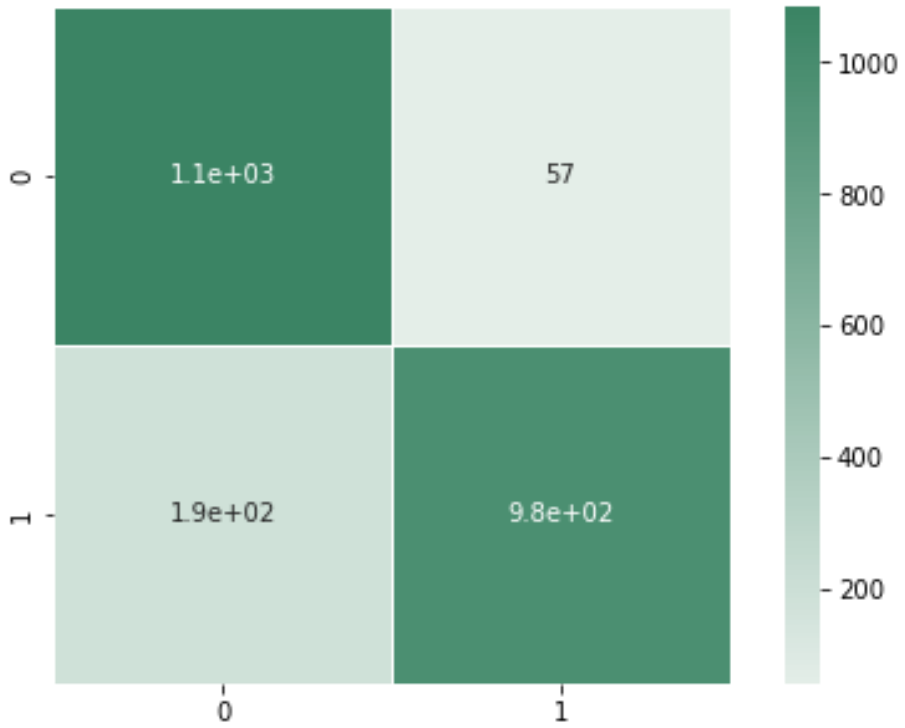

Fig. 5. Confusion matrix for Support Vector Machine with linear kernel.

Multi-Layer Perceptron (MLP).

shuffle=True, hidden_layer_sizes=(100,5). model = MLPClassifier(shuffle = True, hidden_layer_sizes = (100,5))), Test Accuracy = 87.51%

Table 6. Classification Report for Multi-Layer Perceptron.

Class	Precision	Recall	F1-score	Support
0	0.90	0.90	0.88	1142
1	0.85	0.85	0.87	1163

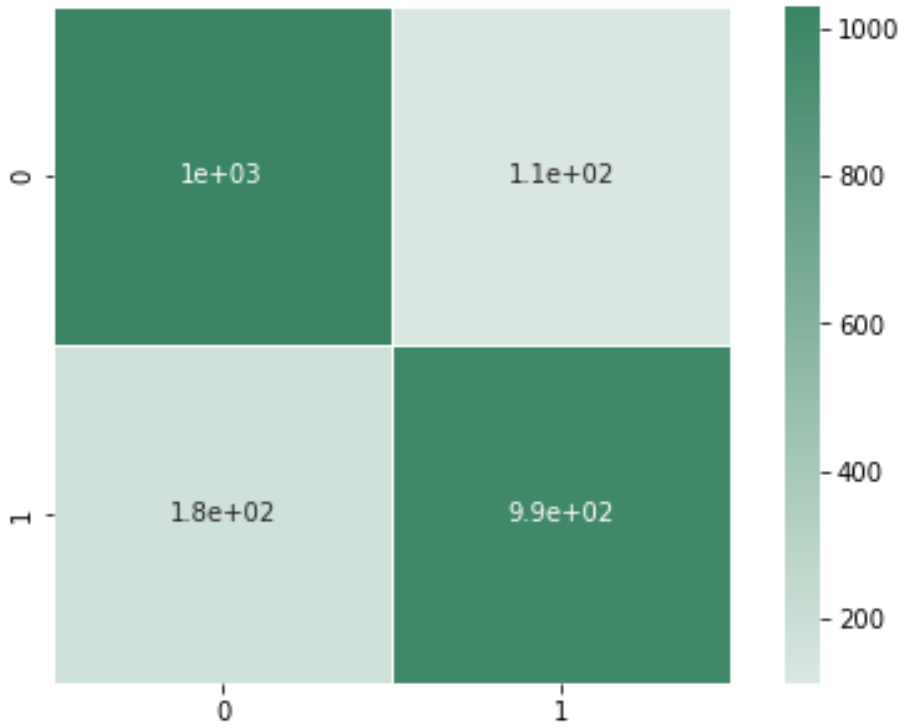

Fig. 6. Confusion matrix for Multi-Layer Perceptron.

Multinomial Naive Bayes.

All default values. model = model = MultinomialNB(), Test Accuracy = 80.31%

Table 7. Classification Report for Multinomial Naive Bayes.

Class	Precision	Recall	F1-score	Support
0	0.88	0.72	0.79	12541
1	0.75	0.89	0.81	11832

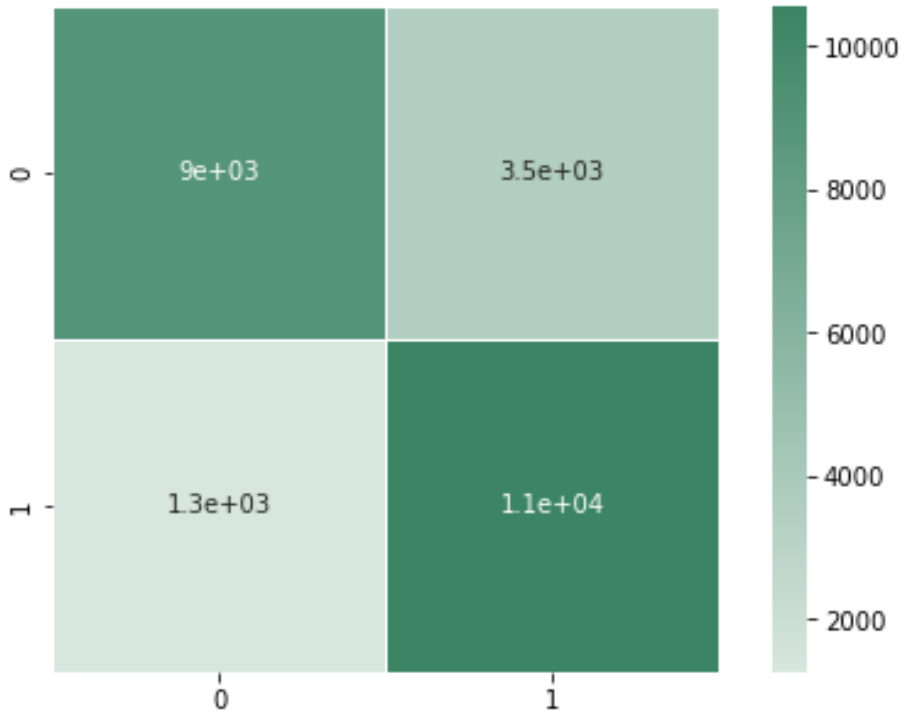

Fig. 7. Confusion matrix for Multinomial Naive Bayes.

7 Comparative Study of the Models

To compare the above-implemented models and choose the best one, along with accuracy, we will check the f1-score more importantly, especially the precision. In our case, false positive cases are more important as it is a significant problem if we identify a comment as spam that is not spam. So to choose the model we will give more importance to Precision rather than normal accuracy.

Table 8. Comparison of the models on Precision.

Model	Precision (Class 0)	Precision (Class 1)
LogisticRegression()	0.84	0.95
LogisticRegression(C=2.0, max_iter=200)	0.85	0.95
SVC()	0.84	0.96
SVC(kernel = ‘linear’)	0.85	0.94
MLPClassifier(shuffle = True, hidden_layer_sizes = (100,5))	0.90	0.85

Table 9. Comparison of the models on Accuracy.

Model	Accuracy
LogisticRegression()	88.81%
LogisticRegression(C=2.0, max_iter=200)	89.33%
SVC()	89.37%
SVC(kernel = 'linear')	89.41%
MLPClassifier(shuffle = True, hidden_layer_sizes = (100,5))	87.51%

We can see from Table 8 that the precision score for the Multi-Layer Perceptron model for Class 1 is lowest at 85%, indicating that the model incorrectly classifies a Genuine comment as Spam 15% of the time.

The precision scores for the Support Vector Machine and Logistic Regression models are very similar and these models suffer less from the problem described in the point before. Among these two models, the accuracy for the Support Vector Machine with the kernel parameter set as "linear" is higher. Hence the Support Vector Machine with the kernel parameter set as "linear" is chosen.

8 Conclusion

This model can easily be implemented as a module in an Application Programming Interface or API to make predictions on various text data sets.

Since the model is made in such a way that it already supports data from any sources, hence only the GUI component needs to be built to support any other platforms like Facebook, X (formerly Twitter) etc.

The model can be deployed into a platform so that the back-end API can make real-time spam predictions to prevent the spam comment from even being posted.

9 Acknowledgement

I would like to acknowledge our guide, Prof. Siladitya Mukherjee, for giving us the opportunity to work under his guidance. We would like to acknowledge YouTube as a social media platform for making the comments of its users publicly available for use in our project as well as UCI Machine Learning Repository for compiling these comments into a single dataset.

References

1. Kantchelian, Alex & Ma, Justin & Huang, Ling & Afroz, Sadia & Joseph, Anthony & Tygar, J.. (2012). Robust detection of comment spam using entropy rate. Proceedings of the ACM Conference on Computer and Communications Security. 59-70. 10.1145/2381896.2381907.
2. Tan, Enhua & Guo, Lei & Chen, Songqing & Zhang, Xiaodong (Frank) & Zhao, Yihong. (2013). UNIK: Unsupervised social network spam detection. International Conference on

Information and Knowledge Management, Proceedings. 479-488. 10.1145/2505515.2505581.

3. Tan, Enhua & Guo, Lei & Chen, Songqing & Zhang, Xiaodong (Frank) & Zhao, Yihong. (2012). Spam Behavior Analysis and Detection in User Generated Content on Social Networks. Proceedings - International Conference on Distributed Computing Systems. 305-314. 10.1109/ICDCS.2012.40.
4. Bratko, Andrej & Cormack, Gordon & Filipic, Bogdan & Lynam, Thomas & Zupan, Blaz. (2006). Spam Filtering Using Statistical Data Compression Models. Journal of Machine Learning Research. 6. 2673-2698.
5. Alberto,T.C. & Lochter,J.V.. (2017). YouTube Spam Collection. UCI Machine Learning Repository. https://doi.org/10.24432/C58885.
6. Almeida,Tiago. (2012). SMS Spam Collection. UCI Machine Learning Repository. https://doi.org/10.24432/C5CC84.
7. "Sklearn.Linear_Model.Logisticregression". https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html#sklearn.linear_model.LogisticRegression.
8. "Sklearn.Svm.SVC". https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html.
9. "Classification: Precision And Recall | Machine Learning | Google Developers". https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall.