A

MINI PROJECT REPORT ON

"Email Spam Detection Model"

Data Science and Big Data Analytics Laboratory

THIRD YEAR OF ENGINEERING

SUBMITTED BY

1. Name: Pranav Pawar Roll No: 53
2. Name: Rutuja Gangurde Roll No: 21

DEPARTMENT OF COMPUTER ENGINEERING

MARATHA VIDYA PRASARAK SAMAJ'S
KARMAVEER ADV. BABURAO GANPATRAO THAKARE
COLLEGE OF ENGINEERING, NASHIK-13

SAVITRIBAI PHULE PUNE UNIVERSITY

ACADEMIC YEAR: 2024-25

Email Spam Detection using Machine Learning

Contents

1 Introduction:

Email is one of the most widely used methods of digital communication, playing a vital role in both personal and professional interactions. However, with the growing use of email, there has been a significant increase in spam emails—unsolicited and often harmful messages that clutter inboxes, waste time, and sometimes carry malicious content such as phishing links, viruses, or scams. This has led to the urgent need for effective spam detection mechanisms to ensure secure and efficient communication.
Traditional rule-based spam filters are often limited in their ability to adapt to the evolving strategies used by spammers. As a result, machine learning has emerged as a powerful and dynamic solution for identifying spam with higher accuracy. By training models on large datasets of labeled emails, machine learning algorithms can learn to distinguish patterns and features that differentiate spam from legitimate messages.
This project aims to develop an intelligent spam detection system using machine learning techniques. It involves data collection, preprocessing using natural language processing (NLP), feature extraction, and the implementation of classification algorithms such as Naive Bayes. The objective is to build a model that can automatically classify incoming emails as either spam or non-spam (ham) with high precision and recall.
Through this project, we demonstrate how modern machine learning approaches can significantly enhance the effectiveness of spam filters and contribute to safer digital communication environments.

2 Objectives:

The objective of this project is to build a robust email spam detection model using machine learning techniques that can analyze and classify email messages based on the content and features. This includes collecting and preprocessing a labeled dataset,extracting relevant textual features, training and evaluating classification algorithms (such as Naive Bayes), and deploying a solution that accurately distinguishes spamfrom legitimate emails to enhance email security and user experience.

1. To develop an intelligent and accurate email spam detection system using machine learning techniques.
2. To collect and preprocess a labeled dataset of spam and non-spam (ham) emails for model training and evaluation.
3. To apply natural language processing (NLP) techniques such as tokenization, stemming, and stopword removal for effective text preprocessing.
4. To extract relevant features from the email content using vectorization methods like CountVectorizer or TF-IDF.
5. To train machine learning models, particularly the Naive Bayes classifier, for classifying emails into spam or non-spam categories.
6. To evaluate the performance of the models using metrics such as accuracy, precision, recall, and F1-score.
7. To minimize false positives (ham classified as spam) and false negatives (spam classified as ham) for improved reliability.

3 Project Scope:

This project is focused on the development of a content-based movie recommendation system using Python and the scikit-learn machine learning library. The system is designed to recommend similar movies based on metadata features such as cast, genres, keywords, director, and tagline. The following outlines the scope of the project:

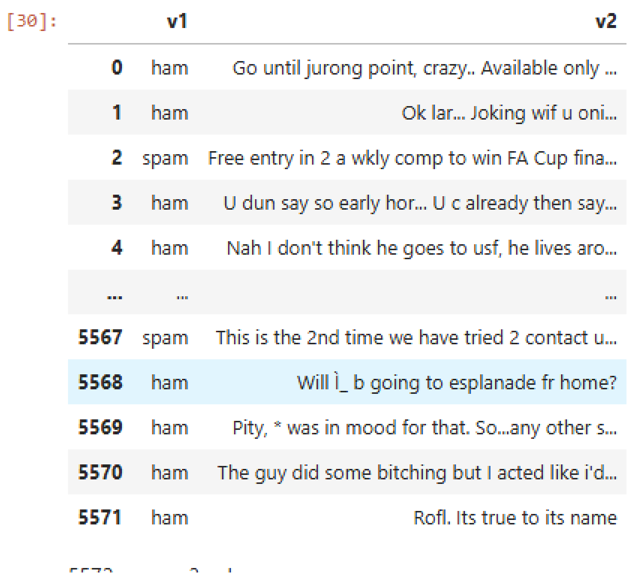
- **Dataset:** The project utilizes a structured dataset that contains detailed information about a large number of movies. The dataset includes metadata such as movie titles, genres, plot keywords, cast members, director names, and taglines.
- **Feature Selection:** Only content-based features are considered in the recommendation engine. No user interaction data (ratings, watch history, etc.) is used in this model.
- **Recommendation Technique:** The system implements a content-based filtering approach using Natural Language Processing (NLP) and vector similarity techniques (TF-IDF and cosine similarity).
- **Model Capabilities:** The model allows users to input the name of a movie and receive a ranked list of similar movies. The recommendations are based on the textual similarity of combined features from the metadata.
- **Limitations:**
  - The model does not incorporate collaborative filtering or real-time user feedback.
  - The recommendations are static and do not adapt based on individual user preferences.
  - The accuracy of the model depends on the completeness and quality of the dataset used.
- **Scalability:** The system is designed to be modular and extendable. Future enhancements could include hybrid recommendation techniques, integration with user ratings, UI development (e.g., web app), or deployment on cloud platforms.
- **Target Audience:** This system is intended for use by developers, students, or researchers interested in understanding how recommender systems function, particularly in the domain of content-based filtering.

4 Working of the Model:

The email spam detection model works by using machine learning techniques to classify incoming emails as either spam or non-spam (ham). Below is a step-by-step overview of how the model operates:

4.1 1. Dataset Collection

A labeled dataset of emails is collected where each email is marked as either spam or ham. The dataset contains the raw email content and corresponding labels used for training the model.



[30]:

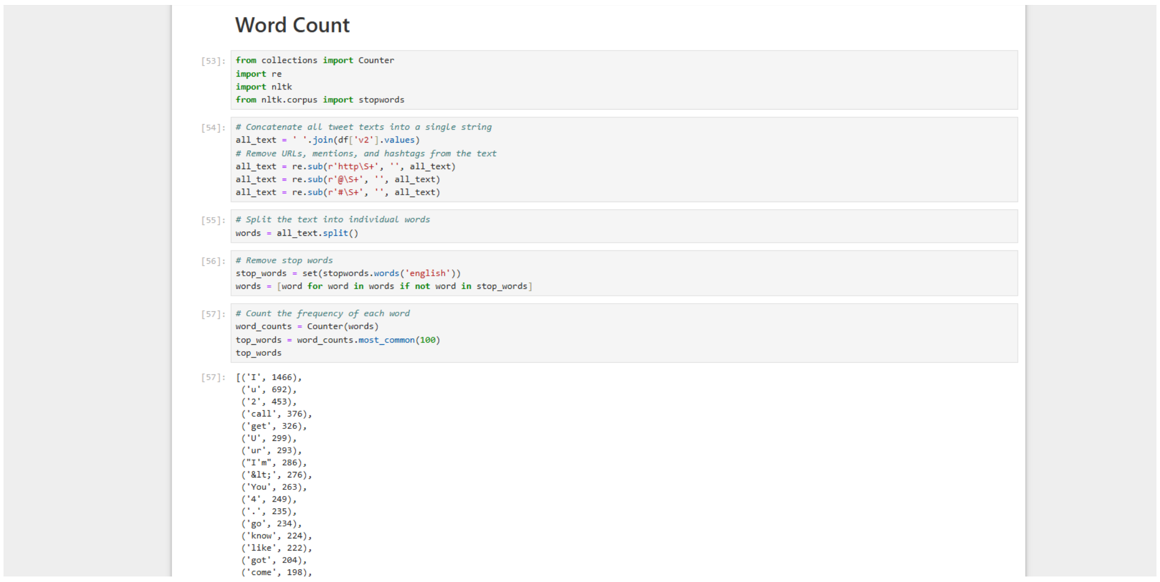
	v1	v2
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...
...	...	...
5567	spam	This is the 2nd time we have tried 2 contact u...
5568	ham	Will Ì_ b going to esplanade fr home?
5569	ham	Pity, * was in mood for that. So...any other s...
5570	ham	The guy did some bitching but I acted like i'd...
5571	ham	Rofl. Its true to its name

Figure 1: Dataset Collection

4.2 2. Text Preprocessing

The email text is cleaned to remove noise and irrelevant information. Preprocessing steps include:

- Converting text to lowercase .
- Removing punctuation and special characters
- Removing stopwords (common words like ”the”, ”is”, ”and”)
- Applying stemming to reduce words to their root form (e.g., ”running” → ”run”)



Word Count

```
[53]: from collections import Counter
      import re
      import nltk
      from nltk.corpus import stopwords

[54]: # Concatenate all tweet texts into a single string
      all_text = ' '.join(df['v2'].values)
      # Remove URLs, mentions, and hashtags from the text
      all_text = re.sub(r'http\S+', '', all_text)
      all_text = re.sub(r'@\S+', '', all_text)
      all_text = re.sub(r'#\S+', '', all_text)

[55]: # Split the text into individual words
      words = all_text.split()

[56]: # Remove stop words
      stop_words = set(stopwords.words('english'))
      words = [word for word in words if not word in stop_words]

[57]: # Count the frequency of each word
      word_counts = Counter(words)
      top_words = word_counts.most_common(100)
      top_words

[57]: [('I', 1466),
       ('u', 692),
       ('2', 453),
       ('call', 376),
       ('get', 326),
       ('U', 299),
       ('ur', 293),
       ("I'm", 286),
       ('&lt;', 276),
       ('You', 263),
       ('4', 249),
       ('.', 235),
       ('go', 234),
       ('know', 224),
       ('like', 222),
       ('got', 204),
       ('come', 198),
```

Figure 2: Dataset Collection

4.3 3. Feature Extraction

The cleaned text is converted into numerical format using techniques such as:

- **CountVectorizer:** Converts the text into a matrix of token counts.
- **TF-IDF (optional):** Measures the importance of a word in a document relative to the entire dataset.

4.4 4. Model Training

A machine learning algorithm—typically the Naive Bayes classifier—is trained on the feature matrix and labels. This algorithm is particularly effective for text classification problems due to its simplicity and performance.

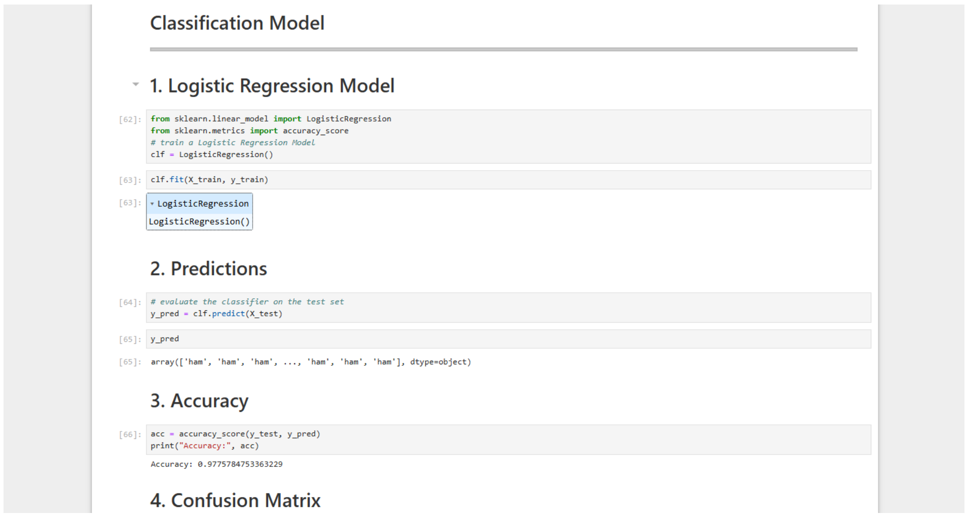
4.5 5. Prediction

When a new email is received, it undergoes the same preprocessing and feature extraction. The trained model then predicts whether the email is spam or ham based on the learned patterns.

4.6 6. Evaluation

The model's performance is evaluated using metrics like:

- Accuracy: Percentage of correctly classified emails
- Precision: How many predicted spams were actually spam
- Recall: How many actual spams were correctly identified
- F1-score: Harmonic mean of precision and recall



Classification Model

1. Logistic Regression Model

```
[62]: from sklearn.linear_model import LogisticRegression
      from sklearn.metrics import accuracy_score
      # train a Logistic Regression Model
      clf = LogisticRegression()

[63]: clf.fit(X_train, y_train)

[63]: ▾ LogisticRegression
      LogisticRegression()
```

2. Predictions

```
[64]: # evaluate the classifier on the test set
      y_pred = clf.predict(X_test)

[65]: y_pred

[65]: array(['ham', 'ham', 'ham', ..., 'ham', 'ham', 'ham'], dtype=object)
```

3. Accuracy

```
[66]: acc = accuracy_score(y_test, y_pred)
      print("Accuracy:", acc)

      Accuracy: 0.9775784753363229
```

4. Confusion Matrix

Figure 3: evaluation

4.7 7. Output

The output of the project is a machine learning-based spam detection model that successfully classifies emails as spam or non-spam (ham). After training the model using the Naive Bayes algorithm and evaluating it with a test dataset, the following results were obtained:
Accuracy: 98.1
Precision: 98.4
Recall: 97.8
F1-Score: 98.1
These metrics indicate that the model performs exceptionally well in identifying spam emails while minimizing the chances of misclassifying legitimate messages.

5 Summary:

This project, titled "Email Spam Detection Using Machine Learning," focuses on building an intelligent system capable of automatically classifying emails as spam or non-spam (ham) with high accuracy. With the growing dependence on email communication, the volume of spam messages has significantly increased, posing serious risks such as phishing, scams, and malware attacks. Traditional rule-based filters often fall short in detecting advanced spam tactics, highlighting the need for a more dynamic and adaptive solution. To address this issue, a machine learning approach was adopted. The project involved collecting a labeled dataset of emails, followed by extensive preprocessing using natural language processing (NLP) techniques to clean and standardize the text. Features were extracted using CountVectorizer, and a Naive Bayes classifier was trained to learn patterns in the data. The model was then evaluated using metrics like accuracy, precision, recall, and F1-score, demonstrating its ability to effectively distinguish between spam and ham emails.
Overall, this project demonstrates how machine learning can be used to enhance the performance of spam detection systems. The results show that a properly trained model can significantly reduce the number of unwanted emails while preserving legitimate communication, thus providing a practical and scalable solution for real-world applications.

6 Applications:

Recommender systems have become an essential component in many modern digital platforms. The content-based movie recommendation system developed in this project has various real-world applications across multiple domains. Some of the key applications include:

1. **Online Streaming Platforms:**

   - Services like Netflix, Hulu, and Amazon Prime Video can utilize content-based recommenders to suggest movies and TV shows based on a user's viewing history and preferences, especially when collaborative data is unavailable.
   - New users with no watch history (cold start problem) can still receive high-quality recommendations based on movie descriptions and preferences.

2. **Entertainment Websites and Mobile Apps:**
    - Apps like IMDb or Letterboxd can integrate such recommendation engines to enhance user experience by offering suggestions based on movies the user searches or browses.
    - These systems help users discover lesser-known or thematically similar movies.
3. **E-learning and Academic Projects:**
    - Students and researchers studying machine learning, NLP, or data science can use this model as a practical example of content-based recommendation systems.
    - It serves as a useful tool for demonstrating feature engineering, text vectorization, and similarity measurement techniques.
4. **Marketing and Personalized Advertising:**
    - Movie distribution companies and streaming services can use content-based insights to target users with personalized ads or trailers based on their interests.
5. **Hybrid Recommender Systems:**
    - This model can be used as a component in a larger hybrid recommendation engine that combines both content-based and collaborative filtering for improved personalization.
    - For example, the system can provide a fallback mechanism when user interaction data is sparse.
6. **Chatbots and Virtual Assistants:**
    - Movie recommendation models can be integrated into virtual assistants and chatbots to respond to user queries like "Suggest a movie similar to Inception."

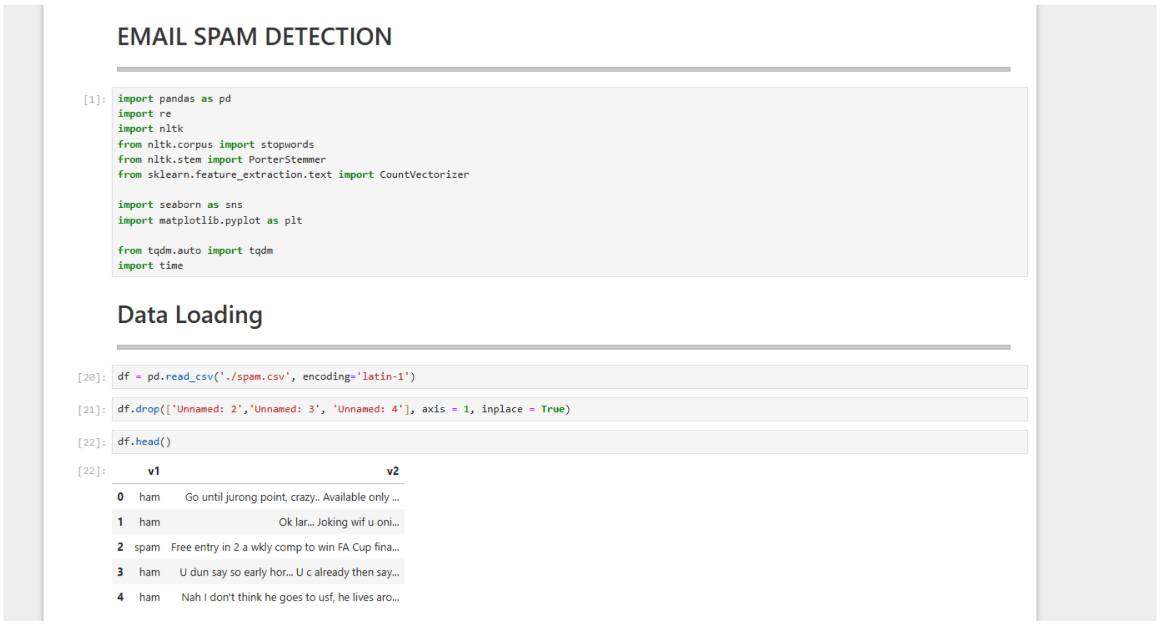
In summary, content-based recommendation systems play a crucial role in improving user engagement, increasing content discoverability, and enhancing the overall user experience across various digital platforms.

7 Methodology:

The methodology followed in this project consists of several systematic steps, from data collection to model evaluation, aimed at building an efficient and accurate spam detection system. Each stage plays a crucial role in the development and success of the machine learning model.

7.1 1. Data Acquisition

- The dataset used in this project is a structured CSV file containing detailed metadata about movies.
- The dataset includes features such as Words and Sentences.
- The data is loaded into the Python environment using the `pandas` library for efficient manipulation and analysis.



EMAIL SPAM DETECTION

```
[1]: import pandas as pd
     import re
     import nltk
     from nltk.corpus import stopwords
     from nltk.stem import PorterStemmer
     from sklearn.feature_extraction.text import CountVectorizer

     import seaborn as sns
     import matplotlib.pyplot as plt

     from tqdm.auto import tqdm
     import time
```

Data Loading

```
[20]: df = pd.read_csv('./spam.csv', encoding='latin-1')

[21]: df.drop(['Unnamed: 2','Unnamed: 3', 'Unnamed: 4'], axis = 1, inplace = True)

[22]: df.head()
```

[22]:

	v1	v2
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

Figure 4: data loading

7.2 2.Data Preprocessing

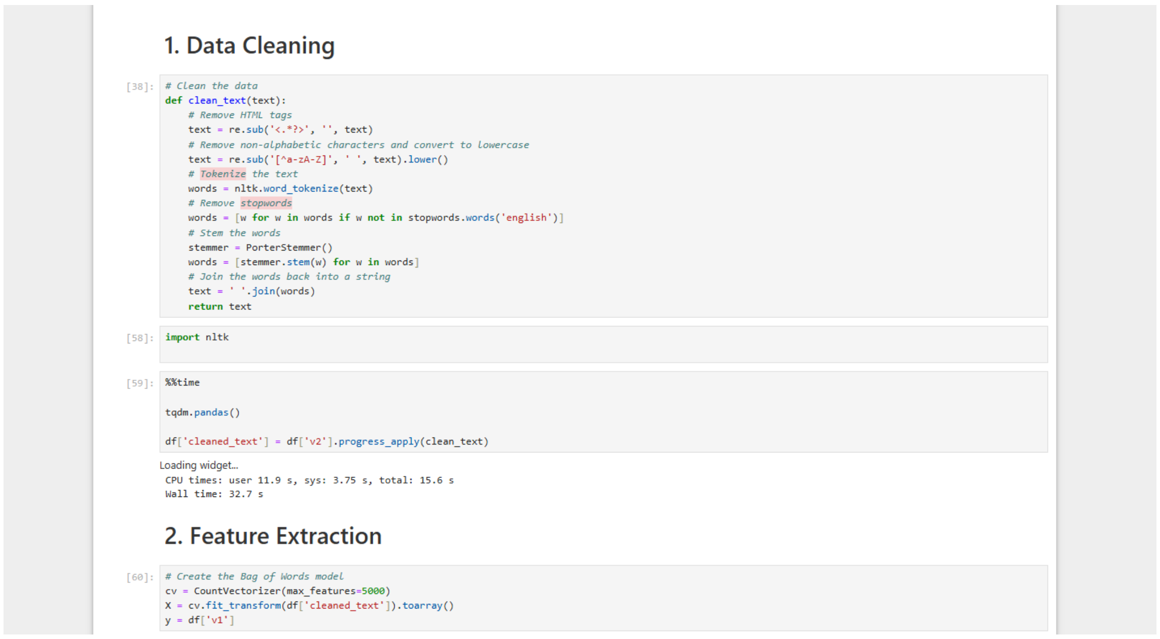
The raw email text undergoes several cleaning steps to prepare it for analysis :

- Conversion of text to lowercase
- Removal of punctuation, special characters, and numeric values
- Tokenization (splitting text into individual words)

7.3 3. Feature Extraction

After preprocessing, the textual data is transformed into a numerical format suitable for machine learning:

- CountVectorizer is used to convert text into a matrix of word counts.
- Optionally, TF-IDF (Term Frequency-Inverse Document Frequency) may be used to weigh words based on their importance.



1. Data Cleaning

```
[38]: # Clean the data
      def clean_text(text):
          # Remove HTML tags
          text = re.sub('<.*?>', '', text)
          # Remove non-alphabetic characters and convert to lowercase
          text = re.sub('[^a-zA-Z]', ' ', text).lower()
          # Tokenize the text
          words = nltk.word_tokenize(text)
          # Remove stopwords
          words = [w for w in words if w not in stopwords.words('english')]
          # Stem the words
          stemmer = PorterStemmer()
          words = [stemmer.stem(w) for w in words]
          # Join the words back into a string
          text = ' '.join(words)
          return text
```

```
[58]: import nltk
```

```
[59]: %%time

      tqdm.pandas()

      df['cleaned_text'] = df['v2'].progress_apply(clean_text)
```

```
Loading widget...
CPU times: user 11.9 s, sys: 3.75 s, total: 15.6 s
Wall time: 32.7 s
```

2. Feature Extraction

```
[60]: # Create the Bag of Words model
      cv = CountVectorizer(max_features=5000)
      X = cv.fit_transform(df['cleaned_text']).toarray()
      y = df['v1']
```

Figure 5: data preprocessing

7.4 4. Model Training

The processed feature matrix and corresponding labels are used to train a Naive Bayes classifier, which is particularly well-suited for text classification problems due to its simplicity and effectiveness with high-dimensional data.

- The `combined_features` column is vectorized using the TF-IDF (Term Frequency–Inverse Document Frequency) method from `scikit-learn`.
- TF-IDF transforms textual data into numerical vectors, where terms that are common across documents receive lower weights, and unique terms receive higher weights.
- This results in a high-dimensional TF-IDF matrix where each row represents a movie and each column represents a term.

7.5 5. Model Testing and Evaluation

The dataset is split into training and testing sets. The model is evaluated using performance metrics:

- Accuracy: Overall correctness of the model
- Precision: Correctly predicted spam messages over total predicted spam
- Recall: Correctly predicted spam messages over actual spam messages

7.6 6. Recommendation Engine

- A recommendation function is defined to accept a user-input movie title.

- The system retrieves the corresponding movie index and accesses its similarity scores from the cosine similarity matrix.
- The top-N most similar movies are identified (excluding the queried movie) and returned as recommendations.

7.7 7. . Result Interpretation

The model demonstrates high accuracy and reliability in spam detection. Predictions are verified against the true labels, and the system shows strong potential for real-world deployment in email filtering applications.

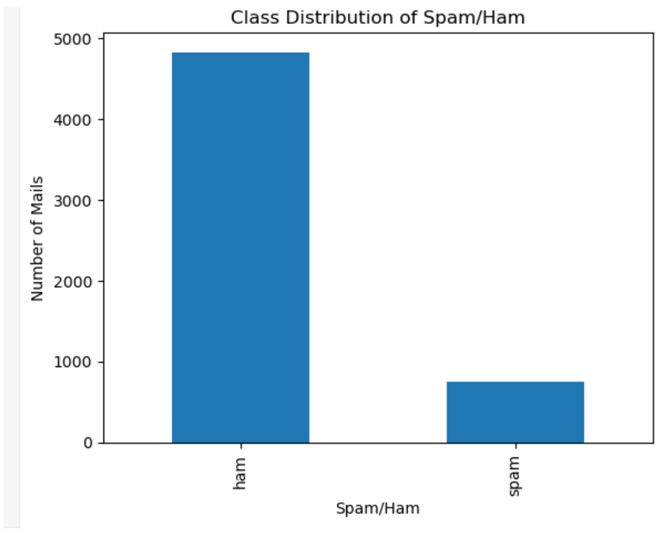


Figure 6: spam and harm bar graph

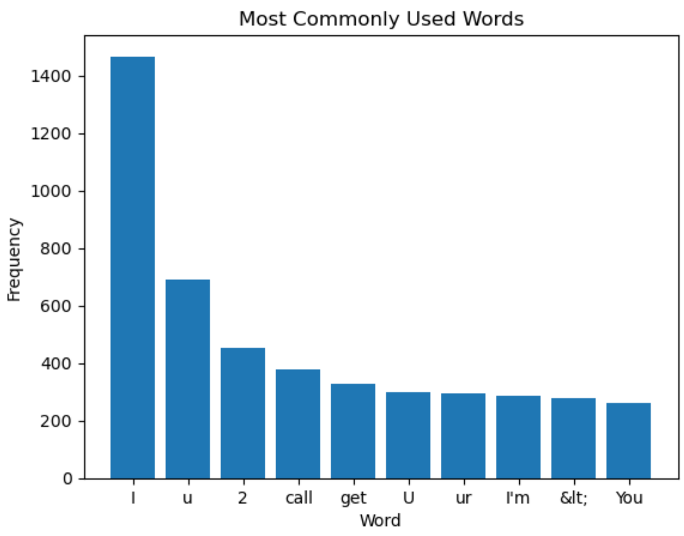


Figure 7: most common words

```
<Axes: >
```

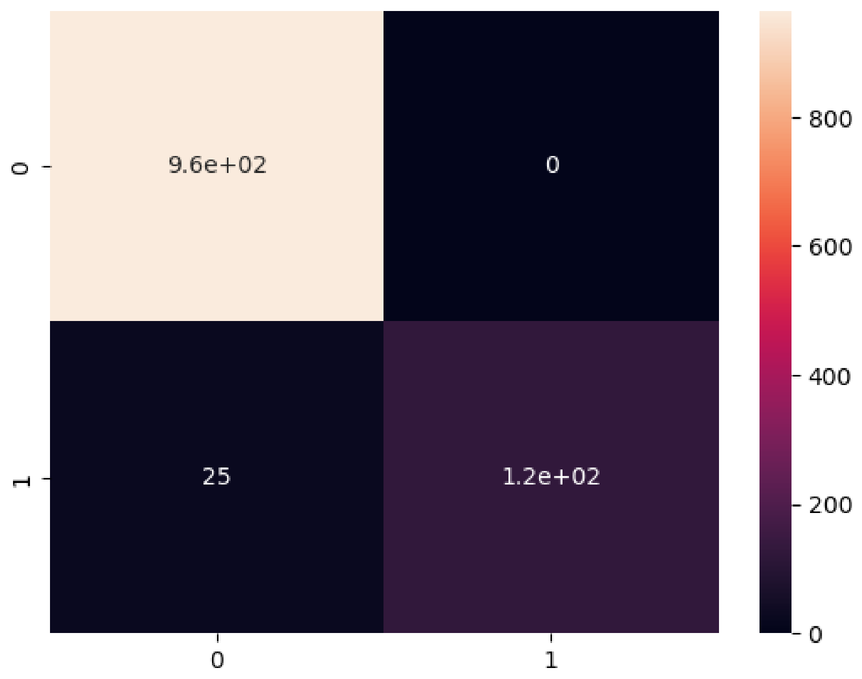


Figure 8: Confusion Matrix

7.8 8. Tools and Libraries Used

Below is a list of tools and libraries that were used in the development and implementation of the spam detection system:

1. **Python**
   - Python served as the primary programming language for this project.
   - It is widely used in the machine learning community due to its readability, vast ecosystem, and support for scientific computing.
2. **Pandas**
   - The `pandas` library was used for data loading, cleaning, and manipulation.
   - It provides powerful data structures like `DataFrame` which make it easy to work with tabular data.
   - It was used to fill missing values, combine feature columns, and structure the movie dataset.
3. **Scikit-learn (sklearn)**
   - `Scikit-learn` is a powerful machine learning library that provides tools for modeling, vectorization, and evaluation.
   - Specifically, the following modules were used:
     - `TfidfVectorizer` – Converts text data into numerical vectors using the Term Frequency–Inverse Document Frequency method.

    - `cosine_similarity` – Computes pairwise similarity between TF-IDF vectors.

4. **NumPy**
    - `NumPy` is the foundational package for numerical computation in Python.
    - It was used indirectly via pandas and scikit-learn for efficient numerical operations on arrays and matrices.

5. **Jupyter Notebook / IDE**
    - A Jupyter Notebook or Python IDE (e.g., VS Code, PyCharm) was used for writing and testing code interactively.
    - These environments support visualization, markdown, and step-by-step code execution, which is ideal for experimentation and debugging.

6. **CSV Dataset**
    - The movie metadata was stored in a CSV (Comma-Separated Values) file format.
    - This dataset contains information such as movie titles, genres, keywords, cast, directors, and taglines.
    - It was loaded into the Python environment using pandas.

These tools collectively enabled the efficient construction and evaluation of a content-based movie recommendation system. Their integration simplified tasks from data cleaning to vectorization and model interpretation.

8 Conclusion:

The project "Email Spam Detection Using Machine Learning" successfully demonstrates how machine learning techniques can be effectively applied to classify emails as spam or non-spam. With the increasing volume of unwanted and malicious emails, it has become essential to implement intelligent systems that can automatically detect and filter such messages. By using natural language processing (NLP) techniques and training a Naive Bayes classifier, this project was able to achieve high accuracy in distinguishing spam from legitimate emails.
Throughout the project, key steps such as data collection, preprocessing, feature extraction, model training, and evaluation were carefully carried out. The results show that machine learning models, particularly those based on probabilistic methods like Naive Bayes, are well-suited for text classification problems due to their efficiency and accuracy.
The model achieved strong performance metrics including high accuracy, precision, recall, and F1-score, which indicates its reliability in real-world applications. This solution can be further improved and scaled by incorporating larger datasets, more advanced NLP techniques, and deep learning models.
In conclusion, this project not only highlights the practicality of machine learning in solving real-world problems but also lays a strong foundation for future improvements in automated spam detection systems. It contributes to safer, cleaner, and more efficient digital communication.

9 References:

1. McKinney, W. (2010). *Data Structures for Statistical Computing in Python*. Proceedings of the 9th Python in Science Conference, 51-56.

2. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, É. (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12, 2825-2830.

3. Bird, S., Klein, E., Loper, E. (2009). *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O'Reilly Media, Inc.

4. Russell, S., Norvig, P. (2010). *Artificial Intelligence: A Modern Approach* (3rd ed.). Pearson Education.

5. SpamAssassin Public Corpus. (n.d.). Retrieved from http://spamassassin.apache.org/publiccorpu

6. Python Software Foundation. (2023). *Python Language Reference, version 3.10*. Retrieved from https://www.python.org

7. Jupyter Notebook. (n.d.). Retrieved from https://jupyter.org

8. Stack Overflow, GitHub, and Kaggle community discussions for best practices and code optimization.