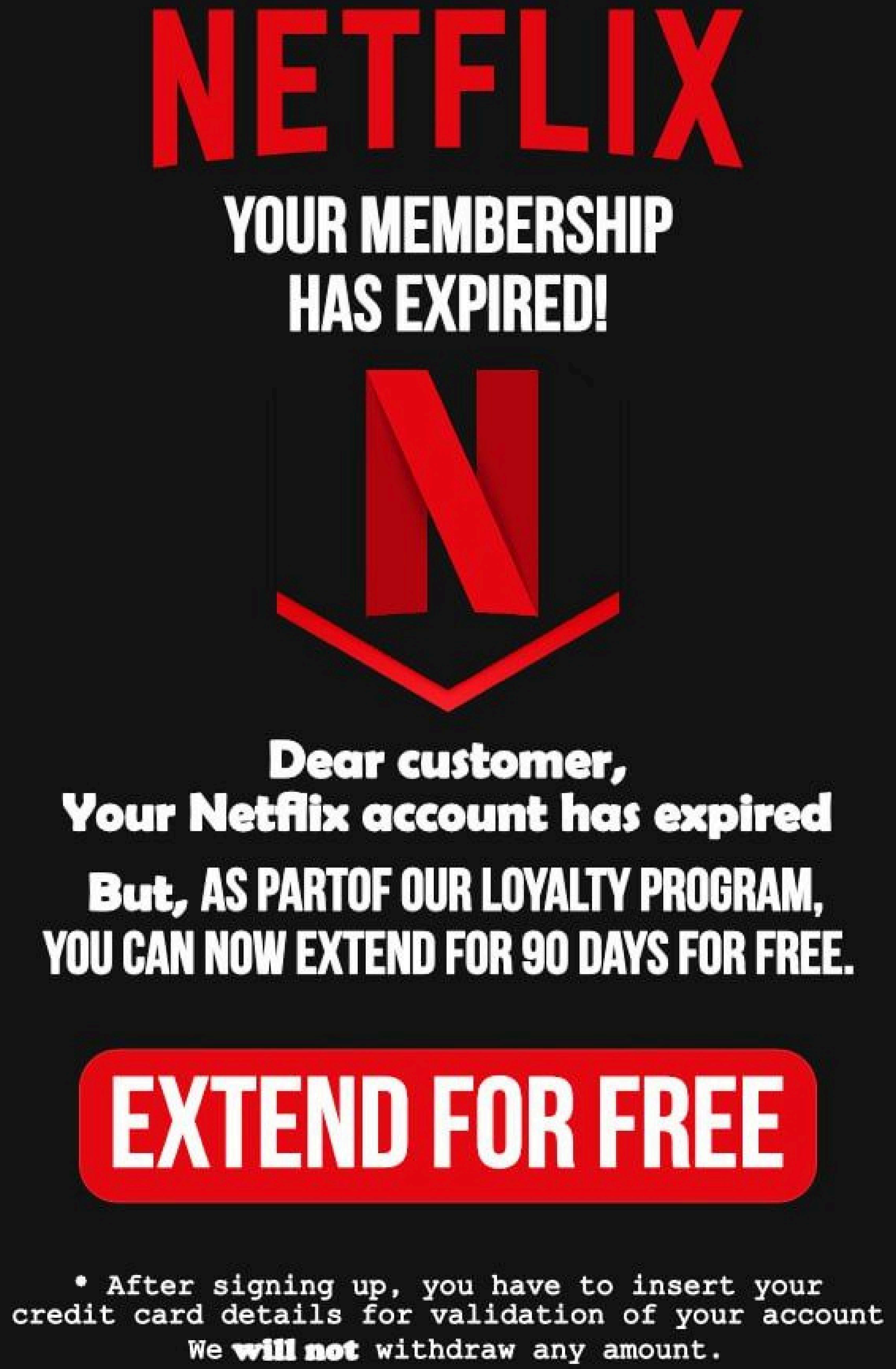
Spam Emails

Ahmed Hamdy



NETFLIX

YOUR MEMBERSHIP
HAS EXPIRED!

Dear customer,
Your Netflix account has expired

But, AS PARTOF OUR LOYALTY PROGRAM,
YOU CAN NOW EXTEND FOR 90 DAYS FOR FREE.

EXTEND FOR FREE

- After signing up, you have to insert your credit card details for validation of your account We **will not** withdraw any amount.

Points of Discussion

These are the broad topics this Report will cover.

01 Introduction to Spam

02 EDA For Spam Emails

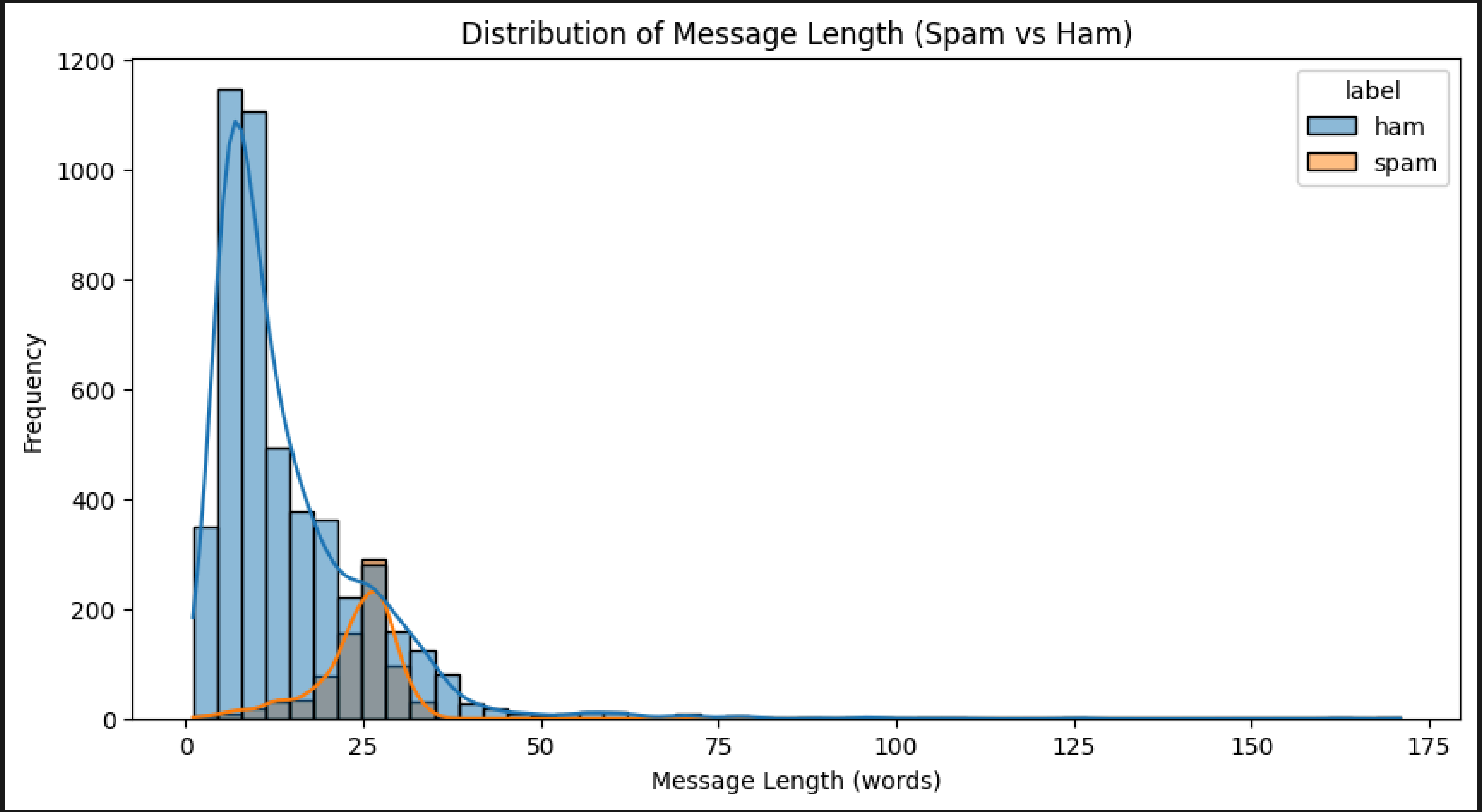
03 The Role of Natural Language Processing in Detecting Spam Emails

04 Models, and Accuracy, Streamlit Cloud

Introduction to Spam

Spam email is unsolicited, bulk electronic messaging, typically sent for commercial gain or malicious intent, that floods recipients' inboxes with advertisements, phishing scams, or malware. Characterized by their unwanted and often irrelevant nature, these messages can compromise personal information, spread viruses, or drive users to harmful websites. The term "spam" itself is a metaphor for the persistent, intrusive nature of these unwanted messages, originating from a Monty Python sketch where the word "Spam" is repeated endlessly.

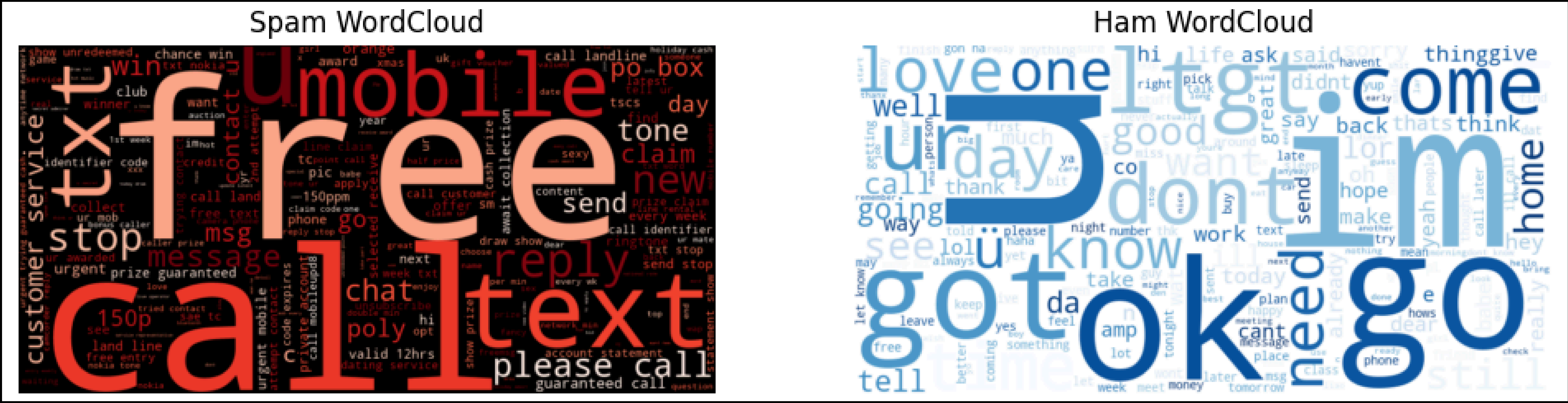
📊EDA For Spam Emails



What is useful to know the length of the message?

Knowing the length of a message is crucial for managing costs, ensuring reliable delivery, and improving readability for the recipient. This is particularly important for text messages (SMS), email marketing, and communication in computer networks.

📊EDA For Spam Emails

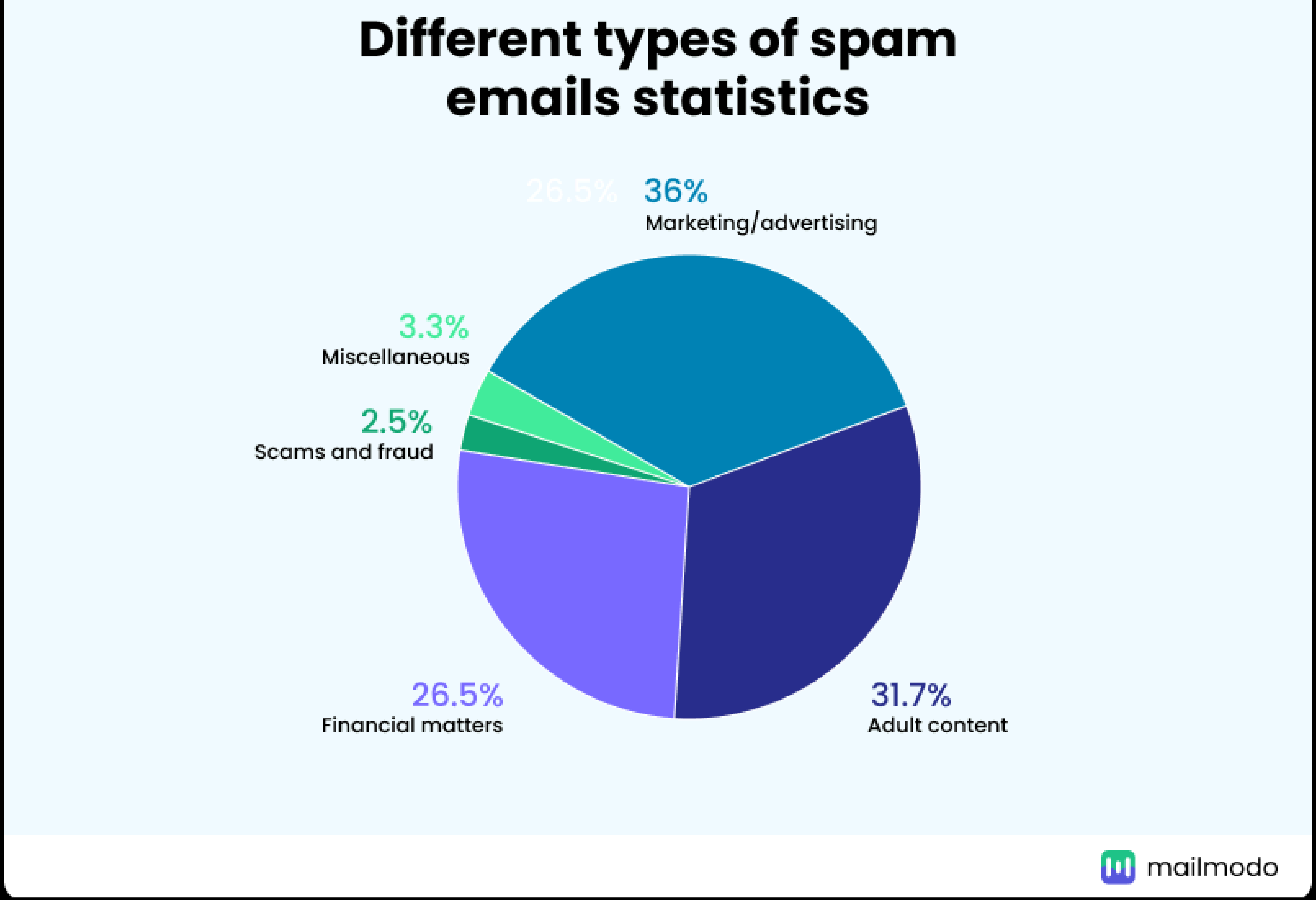


Spam WordCloud

Ham WordCloud

The most common text for spam and Ham

📊EDA For Spam Emails



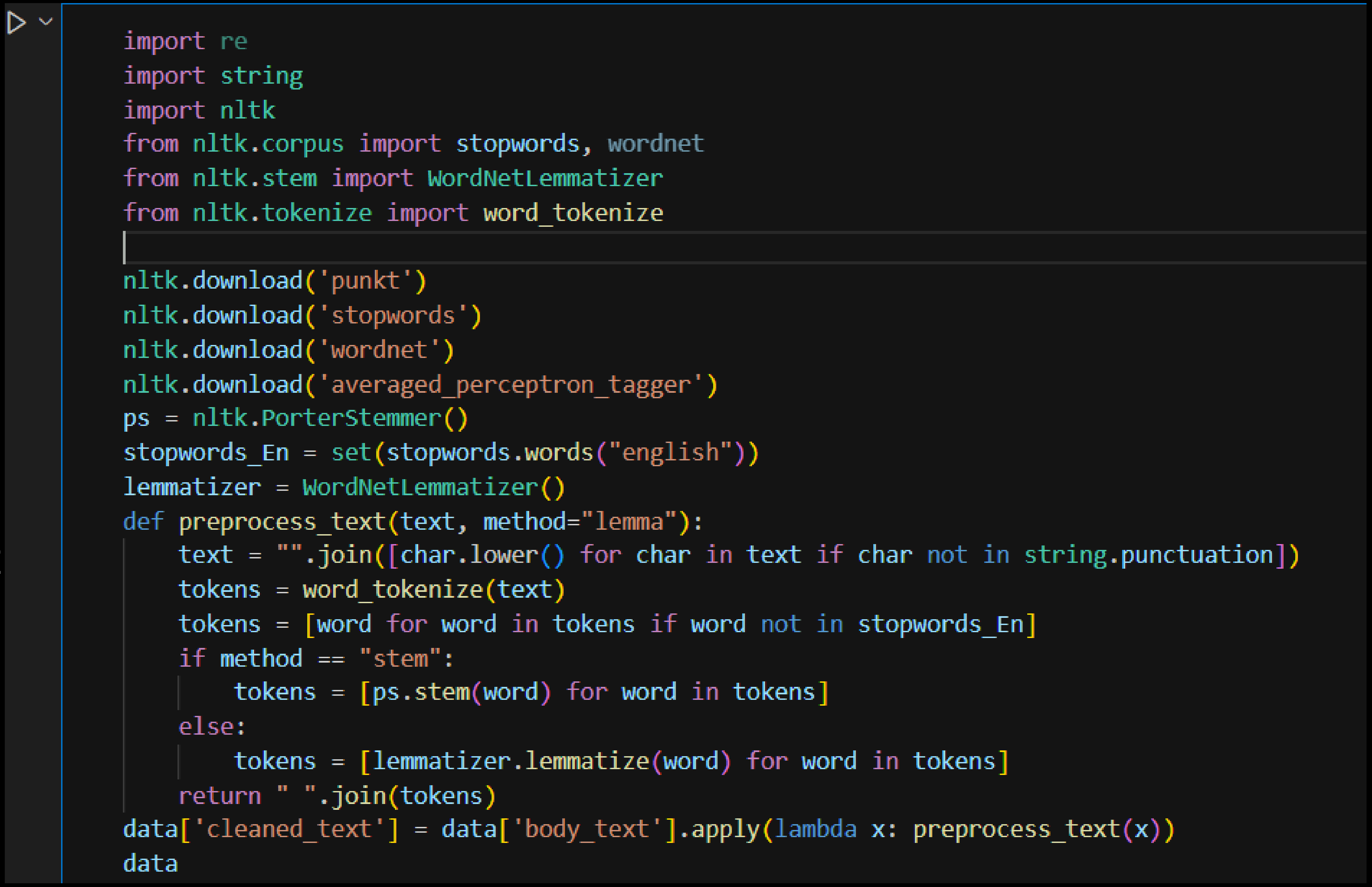
The Role of NLP in Detecting Spam Emails

01 Remove Punctuation

02 Convert it to lowercase

03 Tokenization=>Split text into words

04 Remove Stopwords,and stem

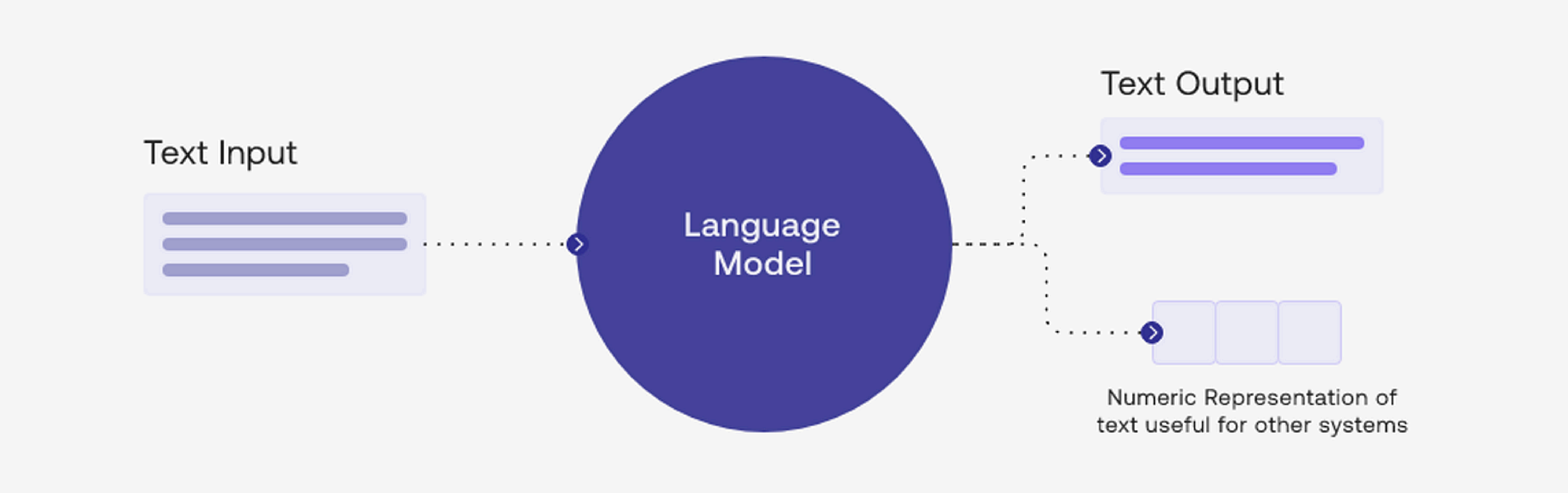


```python
import re
import string
import nltk
from nltk.corpus import stopwords, wordnet
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize

nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger')
ps = nltk.PorterStemmer()
stopwords_En = set(stopwords.words("english"))
lemmatizer = WordNetLemmatizer()
def preprocess_text(text, method="lemma"):
    text = "".join([char.lower() for char in text if char not in string.punctuation])
    tokens = word_tokenize(text)
    tokens = [word for word in tokens if word not in stopwords_En]
    if method == "stem":
        tokens = [ps.stem(word) for word in tokens]
    else:
        tokens = [lemmatizer.lemmatize(word) for word in tokens]
    return " ".join(tokens)
data['cleaned_text'] = data['body_text'].apply(lambda x: preprocess_text(x))
data
```

Create function to remove punctuation, tokenize, remove stopwords, and stem

🤖 NLP Models



01 Naive Bayes: MultinomialNB()

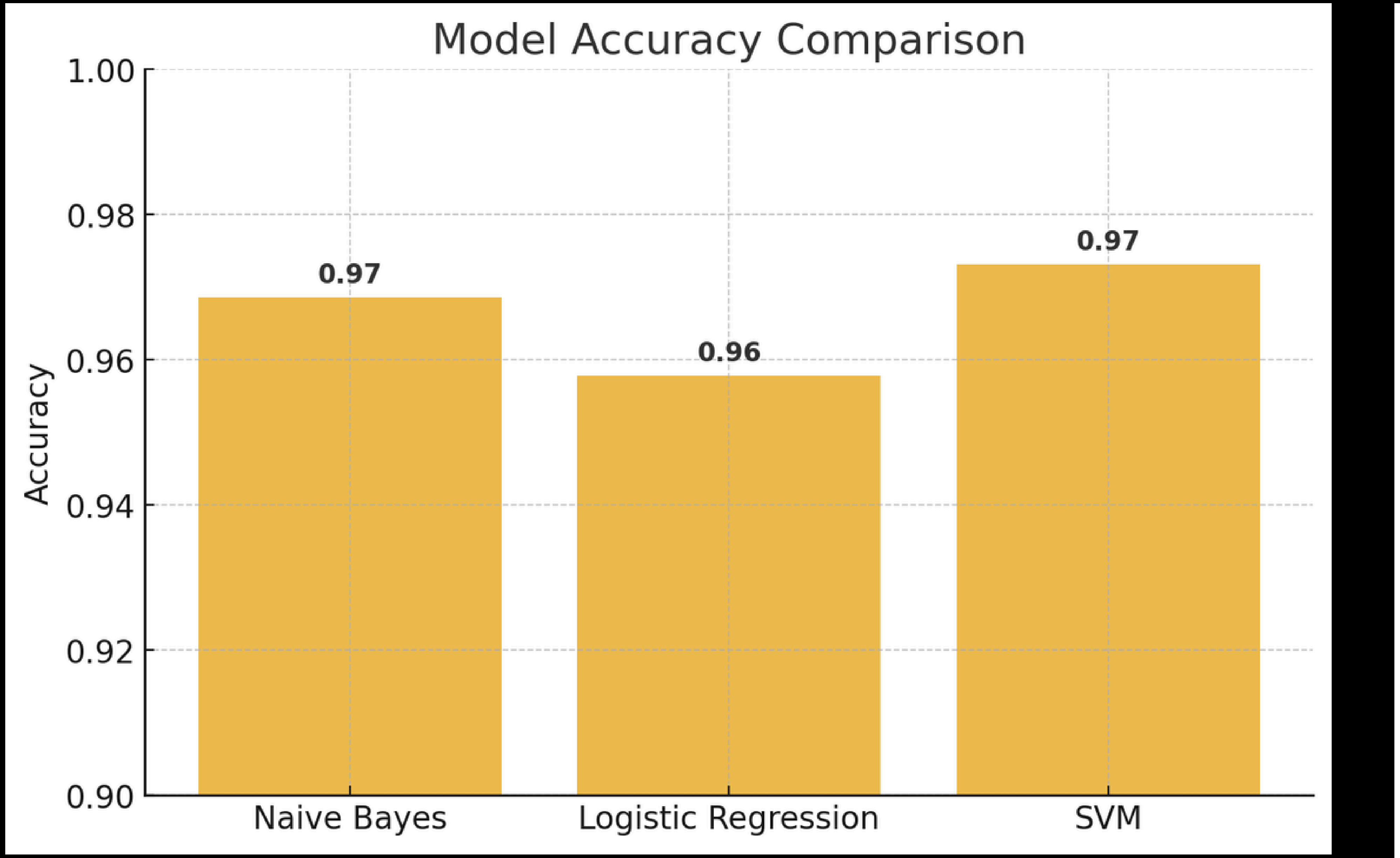
02 Logistic Regression: LogisticRegression(max_iter=200)

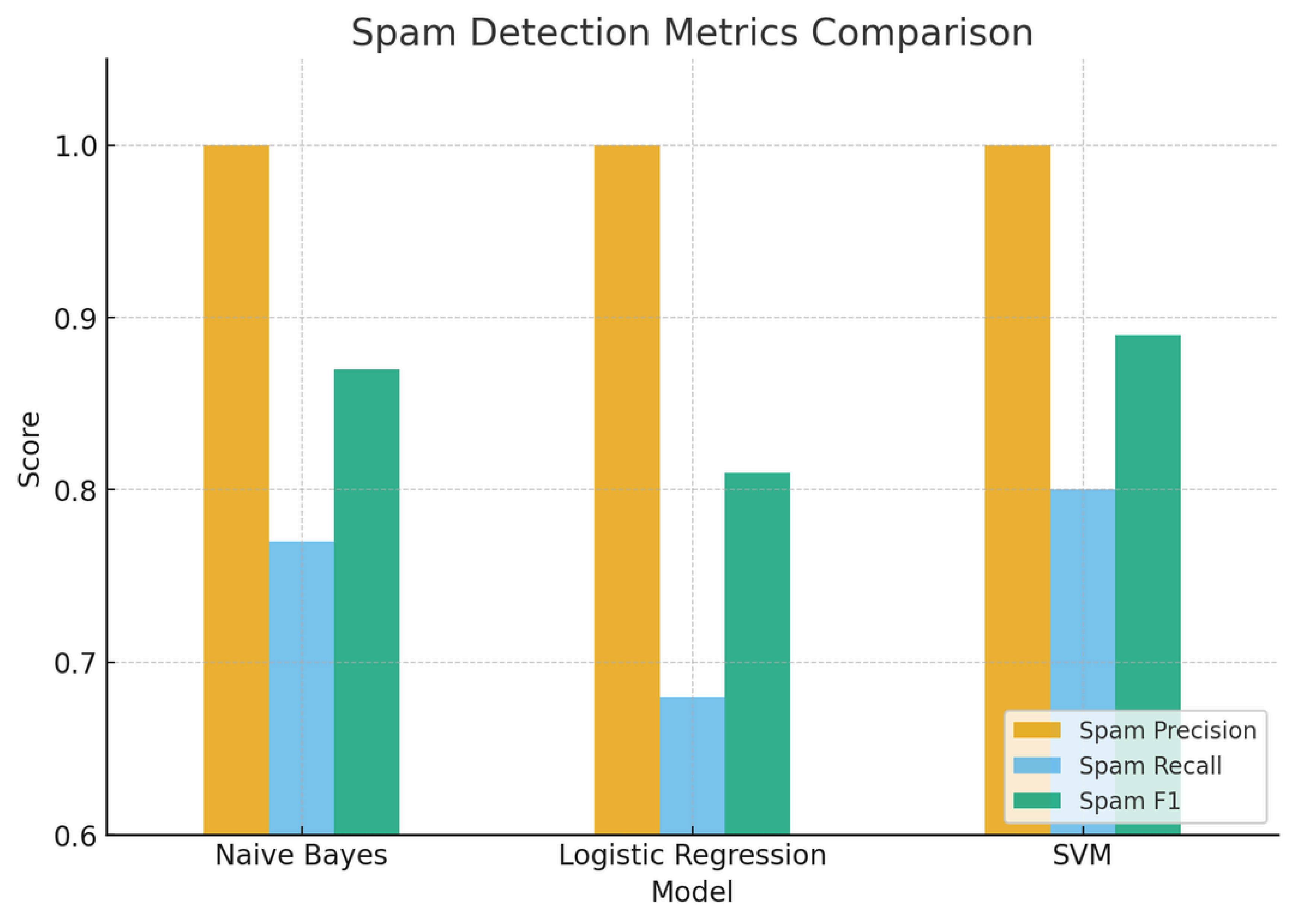
03 "SVM": SVC(probability=True)

📑 Comparison Table

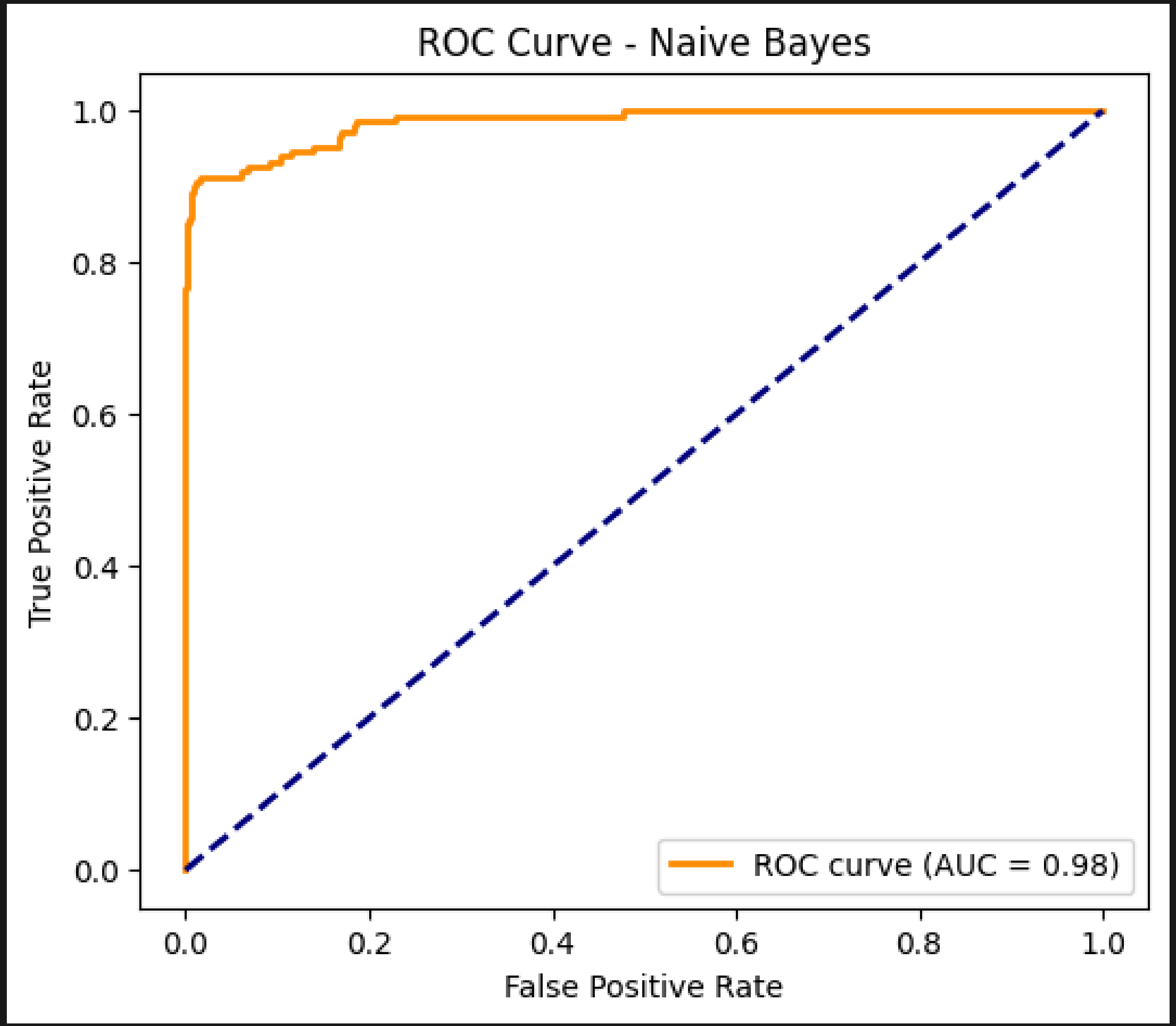
Model	Accuracy	Precision (Ham)	Recall (Ham)	F1 (Ham)	Precision (Spam)	Recall (Spam)	F1 (Spam)	Macro F1	Weighted F1
Naive Bayes	96.86%	0.96	1	0.98	1	0.77	0.87	0.92	0.97
Logistic Regression	95.78%	0.95	1	0.98	1	0.68	0.81	0.89	0.95
SVM	97.31%	0.97	1	0.98	1	0.8	0.89	0.94	0.97

Model Accuracy Comparison





ROC Curve for Naive Bayes



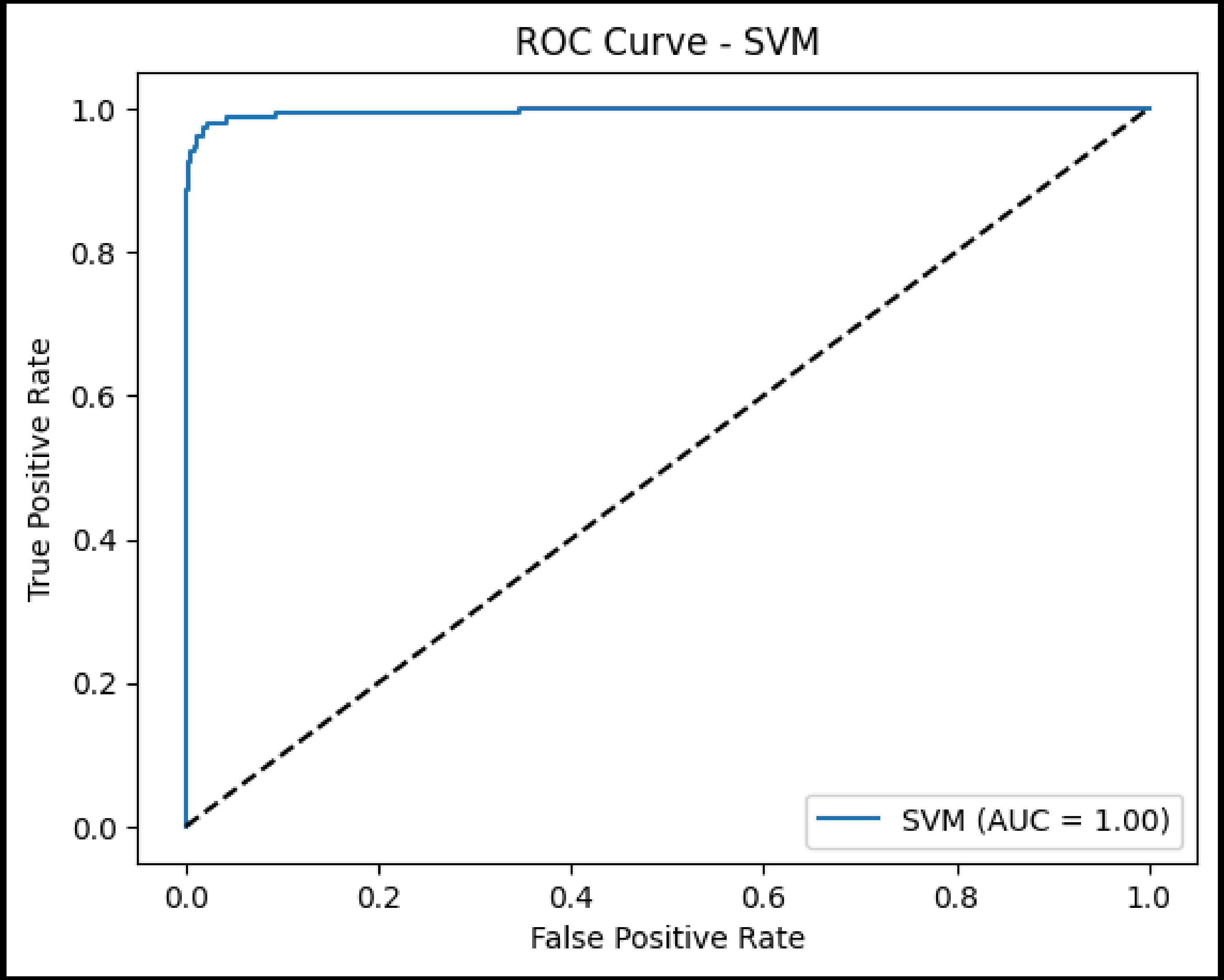
Performs very well with an accuracy of 96.86%.

Strong at detecting ham messages (Recall = 1.00).

Slight weakness in spam detection (Recall = 0.77), meaning it misses some spam messages.

Overall: fast, lightweight, and reliable, making it a great baseline model

ROC Curve for SVM



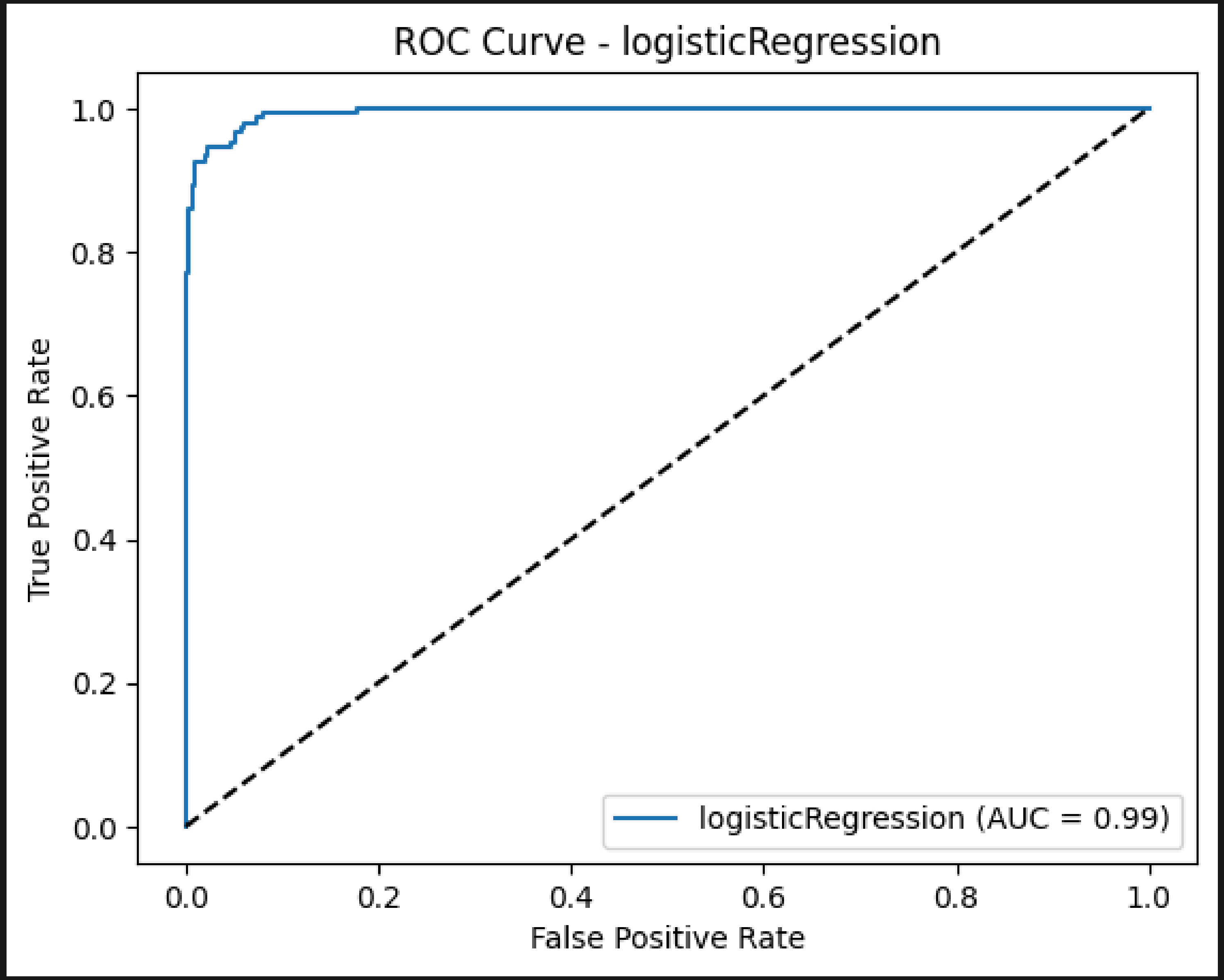
Best performer with 97.31% accuracy.

Maintains a strong balance: ham recall = 1.00, spam recall = 0.80.

Spam precision = 1.00 → very few false positives.

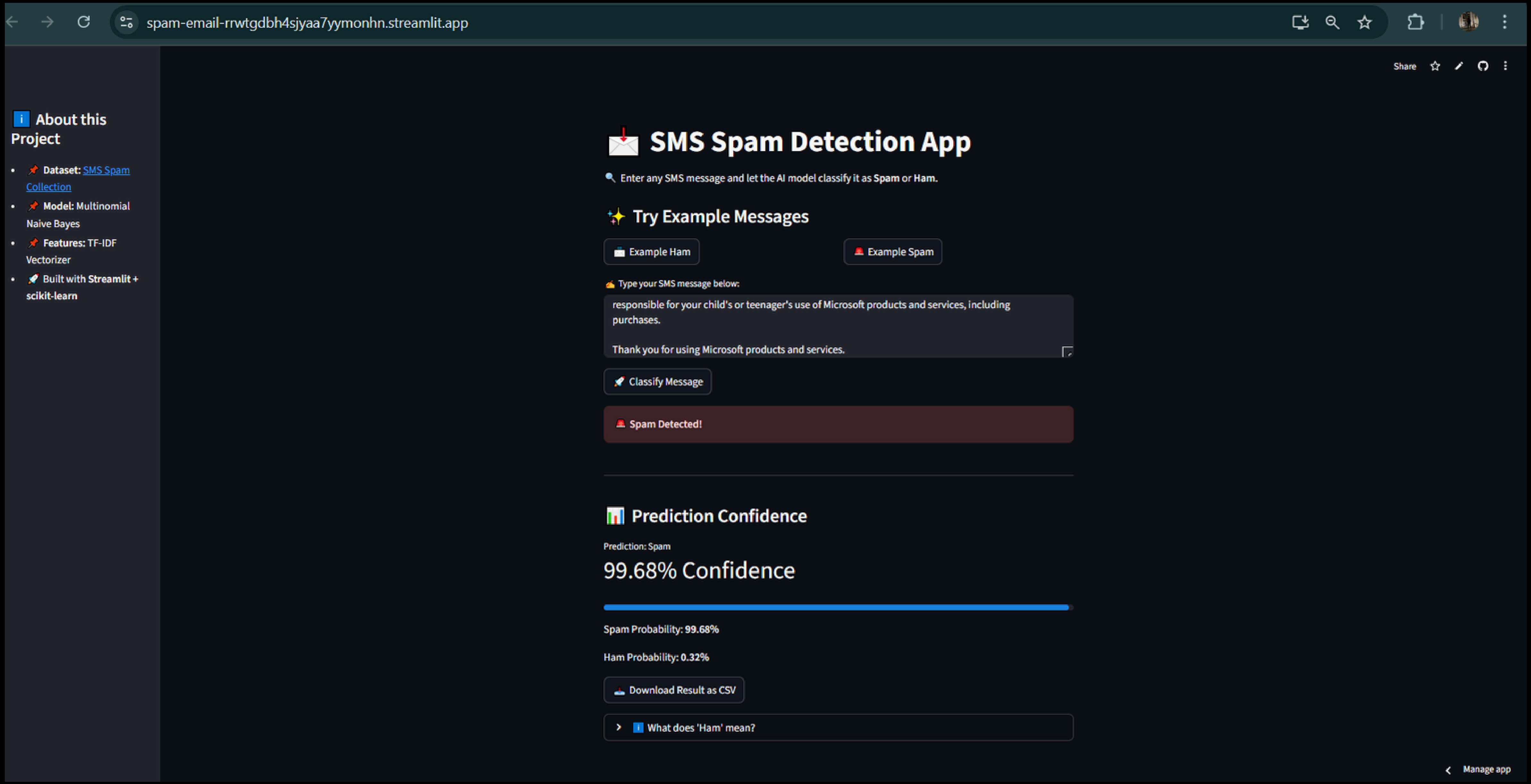
Overall: most robust model, though slower to train on large datasets.

ROC Curve for logistic



Achieves 95.78% accuracy, slightly lower than Naive Bayes and SVM.
Excellent at identifying ham messages (Recall = 1.00).
Struggles with spam recall (0.68) → many spam messages remain undetected.
Good for interpretability, but less effective on imbalanced spam detection

Streamlit cloud



About this Project

- 📌 **Dataset:** SMS Spam Collection
- 📌 **Model:** Multinomial Naive Bayes
- 📌 **Features:** TF-IDF Vectorizer
- 🚀 Built with **Streamlit + scikit-learn**

📩 SMS Spam Detection App

🔍 Enter any SMS message and let the AI model classify it as **Spam** or **Ham**.

✨ Try Example Messages

✍ Type your SMS message below:

responsible for your child's or teenager's use of Microsoft products and services, including purchases.

Thank you for using Microsoft products and services.

🚨 Spam Detected!

📊 Prediction Confidence

Prediction: Spam

99.68% Confidence

Spam Probability: **99.68%**

Ham Probability: **0.32%**

> ℹ What does 'Ham' mean?

Try it live:https://spam-email-rrwtgdbh4sjyaa7yymonhn.streamlit.app/

💡 Advice on Spam Emails

- Never trust unknown senders – avoid clicking links or downloading attachments from suspicious emails.
- Look for red flags – poor grammar, urgent requests, and unfamiliar addresses are common signs of spam.
- Use spam filters – combine machine learning models with built-in email filtering systems for stronger protection.
- Regularly update datasets and models – spam tactics evolve quickly, so continuous training is crucial.
- Educate users – awareness is as important as technology; informed users are less likely to fall victim.

Conclusion

Spam emails remain a major challenge in digital communication, often carrying risks such as phishing, scams, and malware. Effective spam detection models are therefore essential to protect users and ensure secure communication.