# Spam Detection System: Intelligent Information Systems Project

Daniel Torres Montoya

December 2024

## Contents

# 1 Motivation

Spam messages are a daily annoyance for anyone who uses email or SMS. They waste time, clutter inboxes, and sometimes even attempt to scam users. Automating the process of detecting spam messages can make our lives easier, improving both efficiency and security. The goal of this project is to create a system that can intelligently recognize spam messages and separate them from legitimate ones using machine learning.

# 2 Why This Approach?

Spam messages tend to follow certain patterns. Words like *free*, *win*, and *congratulations* appear far more often in spam than in regular messages. By teaching a machine learning model to identify these patterns, we can create a system that understands the difference between spam and legitimate communication. This approach uses statistical methods to analyze text, giving us a scalable and efficient way to solve the problem.

# 3 Short Explanation

Here's a simplified explanation of how the system works:

- First, we train the system using a labeled dataset of messages (marked as spam or not spam).
- The system learns which words or phrases are commonly associated with spam and which are not.
- For example:
  - Spam example: *"Congratulations! You have won $1000! Click here to claim your prize!"*
  - Non-spam example: *"Hi, are we still meeting tomorrow at 10 AM?"*
- When the system encounters a new message, it analyzes the words in the message and uses probabilities to decide if it's spam or not.

# 4 The Mathematics Behind the Model

## 4.1 TF-IDF Vectorization

To analyze text, the system converts each message into numbers using a method called TF-IDF (Term Frequency-Inverse Document Frequency). This method gives us a way to measure how important each word is to a message. Here's how it works:

- **Term Frequency (TF)**: Measures how often a word appears in a message.

$$TF(w) = \frac{\text{Number of times word } w \text{ appears in the message}}{\text{Total number of words in the message}}$$

- **Inverse Document Frequency (IDF)**: Gives less importance to words that are very common across all messages (like *the* or *and*).

$$IDF(w) = \log\left(\frac{\text{Total number of messages}}{\text{Number of messages containing } w}\right)$$

- **TF-IDF Weight**: The final score for each word is calculated as:

$$TF\text{-}IDF(w) = TF(w) \times IDF(w)$$

This turns the message into a vector of numbers, where each number represents the importance of a word.

### 4.2 Naive Bayes Classifier

The system uses a statistical method called the Naive Bayes classifier to predict whether a message is spam or not. This method is based on probabilities:

$$P(\text{Spam}|\text{Message}) = \frac{P(\text{Message}|\text{Spam}) \cdot P(\text{Spam})}{P(\text{Message})}$$

Here's what each term means:

- $P(\text{Spam})$: The probability of a message being spam (based on the training data).
- $P(\text{Message}|\text{Spam})$: How likely the words in the message are to appear in spam messages.
- $P(\text{Message})$: A normalizing factor that ensures the probabilities add up to 1.

The classifier calculates these probabilities for both classes (spam and not spam) and picks the one with the highest probability.

## 5 How the Model Works in Practice

Let's walk through how the system processes a single SMS message step by step:

1. **Read the SMS**: For example, imagine the message is: *"You have won a free iPhone!"*.
2. **Break Down Words**: The system splits the message into individual words, like this: *"you", "have", "won", "a", "free", "iPhone"*.
3. **Transform the Text with TF-IDF**: At this stage, the system calculates how important each word is:
    - First, it looks at how frequently each word appears in the SMS (this is called **term frequency**, or TF).
    - Next, it checks how common or rare the word is across all messages in the dataset (this is the **inverse document frequency**, or IDF).

- Finally, it multiplies these two values together to create a score for each word, turning the SMS into a numeric vector.

4. **Make a Prediction with Naive Bayes**: Now, the system decides if the message is spam or not:
   - It looks at the words in the SMS and calculates how likely they are to appear in spam messages versus non-spam messages, based on what it learned during training.
   - The system then predicts the class (spam or not spam) with the highest probability.

# 6 Implementation Details

## 6.1 Code Overview

Here's a simplified version of the code used to implement the system:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report

# Load the dataset
data = pd.read_csv('spam.csv', encoding='latin-1')
data = data[['v1', 'v2']]
data.columns = ['label', 'message']
data['label'] = data['label'].map({'ham': 0, 'spam': 1})

# Split into training and test sets
X_train, X_test, y_train, y_test = train_test_split(
    data['message'], data['label'], test_size=0.2, random_state
       =42
)

# Convert text into TF-IDF features
vectorizer = TfidfVectorizer(stop_words='english', max_df=0.7)
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)

# Train the model
classifier = MultinomialNB()
classifier.fit(X_train_tfidf, y_train)

# Evaluate the model
y_pred = classifier.predict(X_test_tfidf)
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

Listing 1: Spam Detection Implementation

## 6.2 Execution Result

After running the spam detection code, the system was evaluated on a dataset of 1,115 messages. Below is a summary of the results with resulting code:

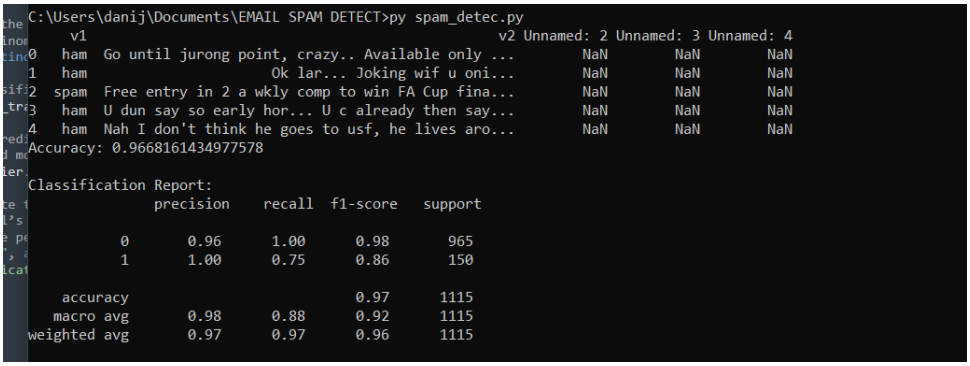


Figure 1: Spam Detection Output

- **Accuracy**: The model achieved an accuracy of **96.7%**, meaning it correctly classified 96.7% of the messages as spam or not spam.
- **Classification Report**:
  - **Precision**: Measures how many messages classified as spam were actually spam. For spam messages (*class 1*), the precision was **100%**.
  - **Recall**: Measures how many actual spam messages were correctly identified. The model achieved a recall of **75%** for spam.
  - **F1-score**: Combines precision and recall into a single metric. The F1-score for spam messages was **86%**.
- **Weighted Metrics**: Taking both spam and non-spam classes into account, the overall weighted precision, recall, and F1-score were all approximately **96%**.

These results demonstrate the model's strong performance in identifying spam messages, while also highlighting areas for improvement, such as recall for spam detection.

# 7 Conclusion

This project demonstrates the power of machine learning in tackling real-world problems like spam detection. By understanding the patterns in spam messages, the system can separate them from legitimate communication with high accuracy. In the future, this approach can be improved further using more advanced natural language processing techniques.

# 8 References


- Kaggle SMS Spam Collection Dataset
- scikit-learn Documentation