# Spam Detection Pipeline Documentation

## Overview

This Python script implements a spam detection pipeline using two machine learning algorithms: Naïve Bayes and Support Vector Machine (SVM). The pipeline processes a text dataset, trains both models, evaluates their performance, and allows users to interactively classify new text inputs as spam or not\_spam. The script includes logging for debugging and performance tracking, and it uses TF-IDF vectorization for text preprocessing.

### Key Features

* Data Preprocessing: Cleans and loads a tab-separated dataset (dataset.csv) with label (spam/ham) and text columns.
* Text Vectorization: Uses TF-IDF to convert text into numerical features, with stop-word removal and n-gram support.
* Model Training: Trains Naïve Bayes (MultinomialNB) and SVM (LinearSVC) models.
* Evaluation: Computes accuracy, precision, recall, and F1-score for both models, with a comparison of results.
* Interactive Mode: Allows continuous user input to classify text as spam or not\_spam, with feature explanations for predictions.
* Logging: Records pipeline progress, timings, and errors to spam\_detection.log and the console.

## Setup Instructions

* Prerequisites:
* - Python 3.8+
* - Required libraries: pandas, numpy, scikit-learn
* - Install dependencies using:

pip install pandas numpy scikit-learn

* Dataset Preparation:
* - The pipeline expects a tab-separated dataset.csv file in the data/ directory with two columns: label (spam or ham) and text.
* - Example dataset format:

ham Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...  
spam Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&C's apply 08452810075over18's

* - Place the dataset at data/dataset.csv. If the file is missing, the script will raise an error.
* Running the Script:
* - Save the scripts as naive\_classifier.py (Naïve Bayes) and svm\_model.py (SVM), or use the combined pipeline.
* - Run either script:

python naive\_classifier.py  
python svm\_model.py

* - The script will:
* - Clean and load the dataset (5,574 samples: 4,827 not\_spam, 747 spam).
* - Train and evaluate the model.
* - Display performance metrics and top features.
* - Enter interactive mode for user input.

## Pipeline Components

### 1. Data Preprocessing

* Cleaning: Removes empty lines, extra tabs, and quotes from the dataset. Saves cleaned data to data/cleaned\_dataset\_nb.csv (Naïve Bayes) or data/cleaned\_dataset\_svm.csv (SVM).
* Loading: Reads the dataset into a Pandas DataFrame, maps ham to not\_spam, and drops invalid rows.
* Output: Logs 5,574 valid samples with class distribution {'not\_spam': 4827, 'spam': 747}.

### 2. Text Vectorization

* Uses TfidfVectorizer with:
* - English stop-word removal.
* - Maximum 20,000 features.
* - N-grams (1 to 3 words).
* - Minimum document frequency of 2.
* Converts training and test text into sparse numerical matrices.
* Logs vocabulary size of 9,977 terms.

### 3. Model Training

* Naïve Bayes: Uses MultinomialNB with smoothing parameter alpha=0.5. Training takes ~0.0045 seconds.
* SVM: Uses LinearSVC with C=1.0, balanced class weights, and maximum 1,000 iterations. Training takes ~0.0138 seconds.
* Logs training completion and time.

### 4. Model Evaluation

* Computes:
* - Accuracy.
* - Precision, recall, and F1-score for the spam class.
* Results from the dataset:

Naïve Bayes Results:

--- Naïve Bayes Results ---  
Accuracy: 0.9776  
Precision (spam): 0.9921  
Recall (spam): 0.8389  
F1-score (spam): 0.9091

SVM Results:

--- SVM Results ---  
Accuracy: 0.9848  
Precision (spam): 0.9521  
Recall (spam): 0.9329  
F1-score (spam): 0.9424

### 5. Feature Analysis

* Logs the top 10 features contributing to spam detection for both models.
* Naïve Bayes: Features with highest spam probabilities:

Top spam features: [('free', 0.0027), ('txt', 0.0019), ('stop', 0.0017), ('text', 0.0016), ('mobile', 0.0016), ('claim', 0.0015), ('reply', 0.0015), ('ur', 0.0014), ('www', 0.0014), ('prize', 0.0013)]

* SVM: Features with highest absolute weights:

Top SVM features: [('uk', 2.2733), ('mobile', 2.1749), ('txt', 1.9388), ('claim', 1.8962), ('150p', 1.7803), ('won', 1.7490), ('50', 1.7381), ('www', 1.6830), ('com', 1.6697), ('video', 1.5602)]

### 6. Interactive Mode

* Allows continuous user input to classify text.
* For each input:
* - Predicts spam or not\_spam using the model.
* - Displays prediction time (~0.001-0.002 seconds).
* - Shows top 5 features influencing the prediction.
* Exits when the user types 'exit' or interrupts with Ctrl+C.
* Handles empty inputs and errors gracefully.

## Demonstration with Example Data

Below are interactions from the SMS Spam Collection dataset, showing predictions for five example texts using both Naïve Bayes and SVM models.

### Example 1: Potential Spam Text

Input:

FreeMsg Hey there darling it's been 3 week's now and no word back! I'd like some fun you up for it still? Tb ok! XxX std chgs to send, £1.50 to rcv

Naïve Bayes Output:

Prediction: not\_spam (Time: 0.0018s)  
Top features: [('50', 0.0009), ('send', 0.0008), ('week', 0.0008), ('word', 0.0004), ('freemsg', 0.0003)]

SVM Output:

Prediction: spam (Time: 0.0019s)  
Top features: [('50', 1.7381), ('std', 0.9097), ('freemsg', 0.8272), ('hey', -0.7746), ('ok', -0.7111)]

Interpretation:

* Naïve Bayes incorrectly predicts not\_spam, possibly due to conversational tone ('hey', 'ok') outweighing spam indicators.
* SVM correctly predicts spam, with strong positive weights for '50', 'std', and 'freemsg' indicating promotional content.

### Example 2: Not Spam Text

Input:

Nah I don't think he goes to usf, he lives around here though

Naïve Bayes Output:

Prediction: not\_spam (Time: 0.0024s)  
Top features: [('don', 0.0002), ('think', 0.0002), ('don think', 0.0001), ('goes', 0.0001), ('lives', 0.0001)]

SVM Output:

Prediction: not\_spam (Time: 0.0016s)  
Top features: [('don', -0.4966), ('nah', -0.2536), ('think', -0.2475), ('goes', -0.1640), ('usf', -0.0944)]

Interpretation:

* Both models correctly predict not\_spam.
* Features like 'don', 'think', and 'nah' have low spam probabilities (Naïve Bayes) or negative weights (SVM), indicating casual conversation.

### Example 3: Spam Text

Input:

Had your mobile 11 months or more? U R entitled to Update to the latest colour mobiles with camera for Free! Call The Mobile Update Co FREE on 08002986030

Naïve Bayes Output:

Prediction: spam (Time: 0.0020s)  
Top features: [('free', 0.0027), ('mobile', 0.0016), ('latest', 0.0005), ('camera', 0.0005), ('colour', 0.0003)]

SVM Output:

Prediction: spam (Time: 0.0020s)  
Top features: [('mobile', 2.1749), ('latest', 0.8945), ('free', 0.8280), ('camera', 0.7550), ('mobiles', 0.3984)]

Interpretation:

* Both models correctly predict spam.
* Key features like 'free', 'mobile', and 'latest' have high spam probabilities (Naïve Bayes) or positive weights (SVM), indicating promotional content.

### Example 4: Not Spam Text

Input:

Is that seriously how you spell his name?

Naïve Bayes Output:

Prediction: not\_spam (Time: 0.0013s)  
Top features: [('seriously', 0.0001), ('spell', 0.0001)]

SVM Output:

Prediction: not\_spam (Time: 0.0019s)  
Top features: [('seriously', -0.1100), ('spell', -0.1066)]

Interpretation:

* Both models correctly predict not\_spam.
* Features 'seriously' and 'spell' have low spam probabilities (Naïve Bayes) or negative weights (SVM), indicating non-promotional content.

### Example 5: Spam Text

Input:

Thanks for your subscription to Ringtone UK your mobile will be charged £5/month Please confirm by replying YES or NO. If you reply NO you will not be charged

Naïve Bayes Output:

Prediction: spam (Time: 0.0019s)  
Top features: [('mobile', 0.0016), ('reply', 0.0015), ('uk', 0.0012), ('ringtone', 0.0007), ('charged', 0.0004)]

SVM Output:

Prediction: spam (Time: 0.0009s)  
Top features: [('uk', 2.2733), ('mobile', 2.1749), ('reply', 1.5506), ('ringtone', 1.4442), ('charged', 0.9049)]

Interpretation:

* Both models correctly predict spam.
* Features like 'uk', 'mobile', and 'reply' have high spam probabilities (Naïve Bayes) or positive weights (SVM), indicating a subscription-based scam.

## Usage Notes

* Dataset: Ensure data/dataset.csv exists and follows the expected format. The dataset used has 5,574 samples (4,827 not\_spam, 747 spam).
* Performance: SVM (98.48% accuracy) outperforms Naïve Bayes (97.76% accuracy), with higher recall (0.9329 vs. 0.8389) and F1-score (0.9424 vs. 0.9091) for spam.
* Feature Explanations: Top features provide insight into predictions. For Naïve Bayes, higher probabilities indicate spam. For SVM, positive weights indicate spam, negative weights indicate not\_spam.
* Error Handling: The script logs errors to naive\_bayes.log or svm.log and handles invalid inputs in interactive mode.

## Limitations

* The dataset is imbalanced (13.4% spam), which may affect Naïve Bayes recall (0.8389). Oversampling or class weighting could improve performance.
* The TF-IDF vectorizer is limited to 20,000 features, resulting in 9,977 terms. Rare terms may be excluded.
* Interactive mode is terminal-based. A GUI would require additional libraries (e.g., tkinter).

## Future Improvements

* Implement cross-validation to improve model robustness.
* Add support for other algorithms (e.g., Random Forest, Neural Networks).
* Develop a web interface for user input.
* Include hyperparameter tuning for Naïve Bayes and SVM.

## License

This project is for educational purposes and provided as-is. The SMS Spam Collection dataset is publicly available for research.

Generated on May 06, 2025