Automated Spam Classification : Project Report

This report details the development and evaluation of a robust machine learning pipeline designed for accurate spam classification, a critical component in maintaining digital communication hygiene.

Our objective is to differentiate between Ham (legitimate) and Spam (unsolicited) messages using Python and the Scikit-learn ecosystem, leveraging foundational techniques in Natural Language Processing (NLP) and binary classification.

Date: November 24, 2025

Author: Jayant Patel

Registration Number: 25BCG10024

Project: Automated Spam Classification

Project Goal: Building a Robust Spam Classifier

The primary goal is to create an effective and efficient machine learning pipeline for classifying text messages. This involves a four-stage architecture designed for optimal data processing, feature extraction, and model deployment.

High Accuracy

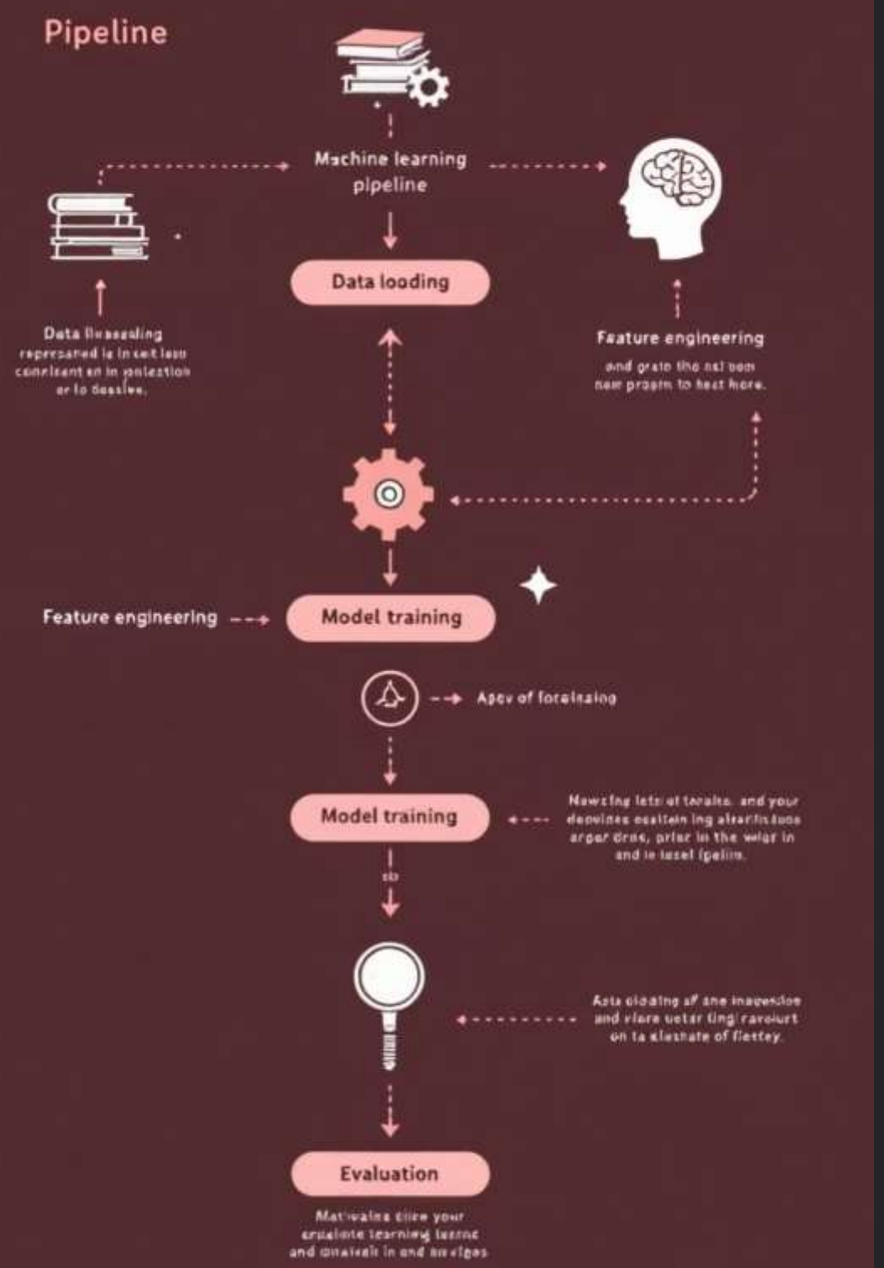
Ensure the model can distinguish between legitimate and spam messages with minimal errors.

Robustness

Develop a system that performs consistently well across diverse datasets and evolving spam tactics.

Scalability

Design a pipeline capable of handling large volumes of text data efficiently.



Technical Architecture: The Four-Stage Pipeline

The classification pipeline is meticulously structured into sequential steps, as defined in the codebase, ensuring a clear and modular approach to machine learning development.

01

Data Loading & Preparation

Initial ingestion, cleansing, and formatting of text data.

02

Feature Engineering (TF-IDF)

Converting raw text into numerical representations for model consumption.

03

Model Training

Developing and training classification models on prepared features.

04

Model Evaluation & Prediction

Assessing model performance and deploying for live predictions.

Step 0: Data Loading and Preparation

The `load_data` function initiates the pipeline, handling data ingestion and ensuring its readiness for subsequent machine learning stages. This foundational step is crucial for the integrity of the entire classification process.

Input Handling

Loads a CSV file (without a header), renaming columns to `label` and `text`. This standardisation ensures consistency and ease of processing.

Label Encoding

Transforms categorical labels ('ham', 'spam') into numerical format: 'ham' → 0 (Negative Class) and 'spam' → 1 (Positive Class). This conversion is essential as machine learning algorithms require numerical inputs.

Data Splitting

The dataset is intelligently partitioned into an 80% training set and a 20% testing set. Stratified sampling (`stratify=y`) is employed to preserve the original proportion of spam and ham messages in both subsets, preventing biased model training or evaluation.

Step 1: Feature Engineering with TF-IDF

Converting raw text into numerical feature vectors is a critical step for any machine learning model. We achieve this using TF-IDF, a powerful technique managed by `sklearn.feature_extraction.text.TfidfVectorizer`.

Mechanism

TF-IDF (Term Frequency-Inverse Document Frequency) assigns a weight to each word. It prioritises terms that frequently appear within a specific document (high Term Frequency) but are rare across the entire corpus (high Inverse Document Frequency). This weighting scheme effectively highlights words that are highly indicative of either spam (e.g., "urgent", "free", "winner") or ham.

Configuration

The vectorizer is carefully configured with `max_features=5000` to restrict the vocabulary to the top 5000 most informative words. Additionally, `stop_words='english'` is applied to exclude common, non-predictive terms (e.g., "the", "is", "a"), which significantly reduces noise and improves feature relevance.

Why TF-IDF?

TF-IDF is preferred over simple word counts (Bag-of-Words) as it accounts for the importance of a word beyond its mere frequency. This helps in capturing the unique lexical patterns associated with spam and legitimate messages.

Step 2: Model Training - Selecting Our Classifiers

To ensure a comprehensive evaluation, two distinct and widely-used classification models were chosen and rigorously trained on the vectorized data (`X_train_vec`). Each model offers unique strengths for text classification tasks.

Logistic Regression (`train_lr_model`)

A robust, linear model renowned for estimating the probability of a message being spam. Highly valued for its interpretability, it provides clear insights into feature importance. The `solver='liblinear'` is employed for its efficiency in handling small to medium-sized datasets.

Multinomial Naive Bayes (`train_nb_model`)

A probabilistic classifier based on Bayes' theorem, which assumes feature independence. MNB is exceptionally effective and fast for sparse text classification tasks, often serving as a strong and reliable baseline model due to its simplicity and strong performance in NLP.

Step 3: Model Evaluation - Assessing Performance

The performance of both trained models is rigorously assessed on the unseen test set, with a particular focus on metrics crucial for real-world spam filtering scenarios. Understanding these metrics is paramount for deploying an effective spam classification system.

Accuracy

Represents the overall correctness of the model's predictions. While a good general indicator, it can be misleading in imbalanced datasets (e.g., where spam is rare).

Precision (Spam)

Crucial for minimizing False Positives – legitimate emails incorrectly flagged as spam. High precision ensures that valid messages are not unjustly quarantined, which is vital for user satisfaction.

Recall (Spam)

Essential for minimizing False Negatives – actual spam messages leaking into the inbox. High recall means more spam is caught, keeping the user's inbox cleaner.

F1 Score

A balanced measure that considers both precision and recall. It is the harmonic mean of precision and recall, providing a single metric that balances both concerns, especially useful when there's an uneven class distribution.

Model Evaluation & Prediction in Action

Beyond static metrics, the `predict_new_email` function serves as a practical demonstration of the deployment stage, showcasing the pipeline's ability to classify new, incoming text inputs instantly using the best-performing trained model.

Prediction Demonstration

This function simulates real-time usage, taking a new message, vectorizing it using the pre-trained TF-IDF vectorizer, and then passing it through the selected classification model to yield a prediction (Ham or Spam).

This allows for immediate assessment of the model's utility in a live environment, highlighting its responsiveness and practical applicability in filtering unwanted communications effectively.

Code Snippet Example

```
def predict_new_email(text_input, model, vectorizer):
text_vec = vectorizer.transform([text_input])
prediction = model.predict(text_vec)    return "Spam"
if prediction[0] == 1 else "Ham"
```

This function illustrates the seamless integration of feature engineering and model prediction for new data.

Conclusion: A Functional and Reliable Pipeline

The project successfully established a functional and reliable spam classification pipeline. Through the training of both Logistic Regression and Multinomial Naive Bayes models on TF-IDF features, the system provides a robust framework for effectively identifying spam.

This modular design ensures that data preparation, feature engineering, training, and prediction stages are distinct, reusable, and easily maintainable, paving the way for efficient future enhancements.

> ### Ready for Deployment
>
> The pipeline is fully prepared for deployment, requiring only an update to the `data_file` path. It is capable of providing high-accuracy binary classification for both SMS and email content.

Future Enhancements and Considerations

While the current pipeline is robust, continuous improvement is key to staying ahead of evolving spam tactics and optimising performance. Several areas have been identified for future enhancements.

1

Advanced NLP Techniques

Explore more sophisticated NLP methods like Word Embeddings (Word2Vec, GloVe) or contextual embeddings (BERT) for richer feature representations, which could capture semantic nuances missed by TF-IDF.

2

Deep Learning Models

Investigate the use of Recurrent Neural Networks (RNNs) or Transformers, particularly for handling sequential text data, which might yield even higher accuracy for complex spam patterns.

3

Real-time Adaptation

Implement mechanisms for continuous learning, allowing the model to adapt to new spam techniques and patterns in real-time, maintaining high effectiveness over time.

4

Performance Optimisation

Further optimise model inference time and resource usage for large-scale production environments, ensuring scalability and cost-effectiveness.