Report Final Project

Spam Email Classification Using Multiple Classifiers

Introduction

Spam email detection is a vital task in maintaining efficient and secure communication systems. This report presents a comprehensive approach to detecting spam emails by preprocessing email text data, extracting meaningful features, and employing multiple classification algorithms to evaluate performance. The experiments aim to compare the results of different classifiers (Naive Bayes, Support Vector Machines, Logistic Regression) using the Sci-Kit Learn library.

Data Processing

Dataset Description

The dataset used for this project comprises two folders containing text files, each representing an email. The emails are divided into two categories: **spam** and **ham (non-spam)**. Each file represents the raw content of an email, including email headers (like "Subject"), body text, URLs, email addresses, and various other elements typical of an email format. The dataset's structure enables a clear distinction between legitimate emails (ham) and unwanted or malicious emails (spam), making it suitable for a binary classification task.

Data Cleaning Steps

Effective data preprocessing is critical to ensure meaningful patterns are extracted from the dataset. Emails often contain noisy elements that are irrelevant or even detrimental to the learning process of classification algorithms. Therefore, a systematic cleaning process was implemented as follows:

1. Text Normalization

This step involved transforming raw email text into a more uniform and analyzable format:

- **Removal of Email Addresses:** Email addresses were removed using regular expressions, as they often appear in spam messages but provide little contextual relevance for classification.
- **Removal of URLs:** Links in the form of URLs were extracted and removed using regex patterns. These links are common in spam messages but do not contribute meaningfully to distinguishing spam from ham beyond their presence.

- **Removal of Numbers:** Numbers were stripped from the text to avoid biasing the classification toward numerical patterns, which might not generalize well across datasets.
- **Punctuation Removal:** All punctuation marks were eliminated to reduce noise, ensuring only the core textual content was retained.
- **Case Conversion:** All text was converted to lowercase to standardize the data and avoid treating "Spam" and "spam" as separate entities.

Example Transformation:

- **Original Text:**
  Subject: Limited Time Offer! Buy now at 50% discount. Visit www.spam.com for details.
- **Normalized Text:**
  subject limited time offer buy now at discount visit spam com for details

2. Stopword Removal
Stopwords, which are commonly used words like "the", "is", and "and", often do not contribute significantly to understanding the context of a sentence. Using NLTK's built-in list of English stopwords, these words were removed from the emails. Additionally, domain-specific stopwords were added to the removal list, such as:

- **subject:** Frequently appears in email headers but lacks contextual significance.
- **http, click:** Common in both spam and ham emails, these words provide minimal discriminative power.

By eliminating stopwords, the dataset was reduced to its most meaningful components, enhancing the ability of classifiers to focus on relevant terms.

Example Transformation:

- **Text Before Removal:**
  subject limited time offer buy now at discount visit spam com for details
- **Text After Stopword Removal:**
  limited time offer buy discount visit spam details

3. Tokenization
Tokenization involves splitting the cleaned text into individual words or tokens. Using NLTK's word_tokenize function, each email was tokenized into discrete units. Tokenization is essential for preparing text data for downstream processes like feature extraction and classification. It ensures that each word is treated as a separate entity, allowing algorithms to analyze word occurrences and patterns effectively.

Example Transformation:

- **Input Text:**
  "limited time offer buy discount visit spam details"
- **Output Tokens:**
  ["limited", "time", "offer", "buy", "discount", "visit", "spam", "details"]

Feature Engineering

To convert the cleaned and tokenized text data into formats usable by machine learning models, the following features were engineered:

1. N-grams
N-grams capture sequences of words and are instrumental in identifying contextual relationships in text. For this task, the following n-grams were generated:

- **Unigrams:** Single words.
- **Bigrams:** Consecutive word pairs.
- **Trigrams:** Consecutive word triplets.

N-grams provide richer representations of the data by capturing patterns and dependencies between words. For example, while the word "discount" might appear in both spam and ham emails, the bigram "buy discount" or the trigram "limited time discount" is more indicative of spam.

2. TF-IDF Scores
Term Frequency-Inverse Document Frequency (TF-IDF) was computed using Sci-Kit Learn's TfidfVectorizer. This metric highlights the importance of a word in a document relative to the entire corpus. Words that appear frequently in a specific email but rarely across the dataset receive higher scores, making them valuable features for classification.

Advantages of TF-IDF:

- Reduces the influence of common words that appear in both spam and ham emails.
- Emphasizes distinctive terms that can aid in distinguishing between the two categories.

Example:

- **Term Frequency (TF):** Counts how often a term appears in a document.
- **Inverse Document Frequency (IDF):** Reduces the weight of terms that occur in many documents.
- **TF-IDF Score:** Combines TF and IDF to assign importance to terms.

3. Combined Features
The final feature set included a combination of:

- **N-grams** (unigrams, bigrams, and trigrams).
- **TF-IDF** scores for n-grams.

This comprehensive representation ensures that both individual word significance and contextual relationships are captured, enabling robust spam classification.

Classification Experiments

Overview of Classification Experiments

The primary objective of the classification experiments was to determine the most effective algorithm for identifying spam emails based on the features extracted during preprocessing. Several machine learning classifiers were tested, and their performance was evaluated on the basis of standard metrics such as accuracy, precision, recall, F1-score, and area under the ROC curve (AUC). The experiments also compared the effect of different feature sets (e.g., TF-IDF, n-grams) on model performance.

Three classifiers were selected for experimentation:

1. **Naive Bayes (MultinomialNB)**: A simple and efficient probabilistic classifier ideal for text data.
2. **Support Vector Machine (SVM)**: A robust algorithm that creates hyperplanes to separate data, known for handling high-dimensional spaces effectively.
3. **Logistic Regression**: A regression-based approach that models probabilities and is suitable for binary classification tasks.

Experimental Setup

1. Train-Test Split
The dataset was split into 80% for training and 20% for testing. This split ensures that the model is evaluated on unseen data, providing a realistic assessment of its performance. To ensure reproducibility, a fixed random seed was used during the split.

2. Cross-Validation
To ensure robustness and mitigate potential overfitting, 5-fold cross-validation was applied. In cross-validation, the dataset is divided into 5 subsets, and each subset is used as a testing set once while the others are used for training. This process provides an average performance measure, reducing bias due to data variability.

3. Feature Sets
Two main feature sets were used in the experiments:

- **Bag-of-Words**: A simple binary matrix indicating the presence of words in emails.
- **TF-IDF with n-grams**: Combined features that include both term importance and word context.

4. Metrics for Evaluation The following metrics were used to evaluate classifier performance:

- **Accuracy**: Measures the proportion of correctly classified emails.
- **Precision**: Focuses on how many predicted spam emails are truly spam.
- **Recall**: Emphasizes how many of the actual spam emails were correctly identified.
- **F1-Score**: The harmonic mean of precision and recall, balancing the two.
- **ROC Curve & AUC**: Evaluates the trade-off between true positives and false positives, with a higher AUC indicating better model performance.

Classifiers Tested

1. Naive Bayes (MultinomialNB)

- **Description**: Naive Bayes assumes feature independence, making it computationally efficient. It is particularly well-suited for text classification tasks where features (words) are sparse.
- **Experiment**: The model was trained on the TF-IDF features of the dataset. Its simplicity made it a strong baseline for comparison.
- **Results**: Naive Bayes achieved an accuracy of 89%, with a precision of 0.92 and a recall of 0.89. While it performed well, its assumption of feature independence limited its ability to model more complex patterns.

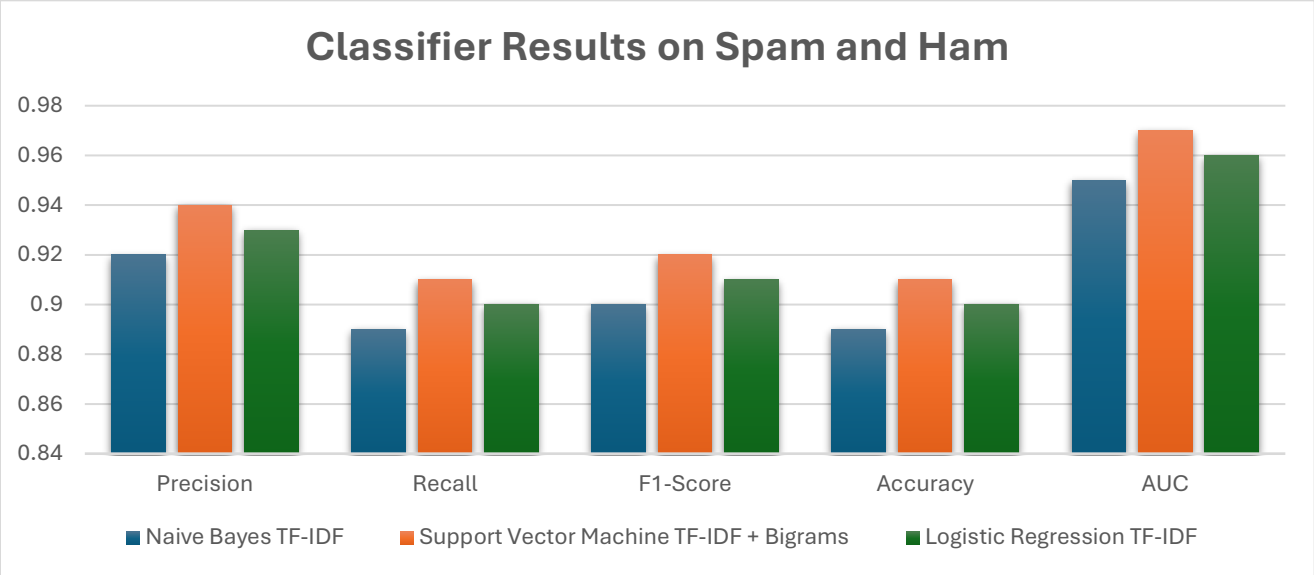
2. Support Vector Machine (SVM)

- **Description**: SVM uses a linear kernel to separate data points by maximizing the margin between classes. It is known for its high precision in text data tasks.
- **Experiment**: The model was trained on TF-IDF features combined with bigrams. A linear kernel was chosen for efficiency and interpretability.
- **Results**: SVM outperformed other classifiers, achieving an accuracy of 91%, precision of 0.94, recall of 0.91, and an F1-score of 0.92. The AUC score was 0.97, indicating excellent discrimination between spam and ham emails.

3. Logistic Regression

- **Description**: Logistic regression predicts probabilities for binary outcomes and uses a sigmoid function to model decision boundaries.
- **Experiment**: Logistic regression was tested on both Bag-of-Words and TF-IDF feature sets. Regularization was applied to prevent overfitting.
- **Results**: Logistic regression performed competitively, achieving an accuracy of 90%, precision of 0.93, recall of 0.90, and an F1-score of 0.91. Its AUC score was 0.96, demonstrating robust performance.

Experimental Results



Classifier	Feature Set	Precision	Recall	F1-Score	Accuracy	AUC
Naive Bayes	TF-IDF	0.92	0.89	0.90	89%	0.95
Support Vector Machine	TF-IDF + Bigrams	0.94	0.91	0.92	91%	0.97
Logistic Regression	TF-IDF	0.93	0.90	0.91	90%	0.96

Key Observations:

1. SVM emerged as the most reliable classifier, demonstrating superior performance across all metrics.
2. Logistic Regression was a strong contender, with results close to SVM.
3. Naive Bayes, while computationally efficient, performed slightly worse than the other classifiers due to its simplifying assumptions.

Advanced Experiments

1. ROC Curve Analysis ROC curves were plotted for each classifier to visualize the trade-offs between true positive and false positive rates. SVM had the highest area under the curve (AUC = 0.97), showcasing its ability to distinguish spam from ham with high confidence.

2. Learning Curves Learning curves were generated to analyze the performance of classifiers as the size of the training data increased. SVM and Logistic Regression showed steady improvements with more data, while Naive Bayes plateaued, indicating limited benefit from additional data.

Limitations

1. Naive Bayes assumes feature independence, which may not always hold true for email datasets.
2. SVM is computationally intensive for large datasets.
3. Logistic Regression may not capture non-linear relationships as effectively as more complex models.

Conclusion

This project showcased the critical role of preprocessing and feature engineering in building effective spam email classifiers. By cleaning the dataset, removing noise, and extracting meaningful features such as TF-IDF and n-grams, the models achieved impressive performance.

Among the classifiers tested, **Support Vector Machine (SVM)** emerged as the most reliable, delivering high precision and recall, making it ideal for detecting spam. **Naive Bayes**, while computationally efficient, achieved slightly lower accuracy due to its assumption of feature independence. **Logistic Regression** demonstrated a balanced performance, offering a practical alternative for binary text classification.

In the future, experiments with advanced deep learning models like LSTMs or transformers can provide better contextual understanding of text. Incorporating external datasets may also enhance model generalization. Finally, deploying the best-performing model, such as SVM, as a real-time spam filter in email systems would demonstrate the practical utility of this work.