CMPUT 466 Mini Project Report

Introduction

Email spam filtering is an important task that helps keep our inboxes free from unwanted or harmful messages. With so many people using email every day, spam emails have become a big problem. These emails often include things like fake offers, scams, or annoying advertisements. In this project, I worked on solving this problem using logistic regression. I used a method called logistic regression to build a model that can tell the difference between spam and regular emails. To do this, I processed the email text using a technique called TF-IDF, which helps the model understand important words in the emails. The goal of this project is to create a system that is good at spotting spam emails quickly and accurately.

Problem Formulation

The objective of this project is to develop a machine learning model to classify emails into two categories: spam and non-spam. The input to the model is the text content of an email, which is processed into a numerical representation using the Term Frequency-Inverse Document Frequency (TF-IDF) method. The output is a binary label, where 1 represents a spam email and 0 represents a non-spam email.

Dataset

The dataset used for this project is sourced from the Kaggle repository: Spam Email Dataset.

It contains a total of 5,729 samples, out of which:

- 1732 emails are labeled as spam.
- 3997 emails are labeled as non-spam.

This dataset is well-suited for binary classification tasks and provides a balanced challenge in terms of addressing class imbalance, where the majority of emails are non-spam. The dataset includes textual data for each email along with corresponding labels, making it ideal for supervised learning.

The problem involves preprocessing the raw text data, building a logistic regression model, and optimizing it to achieve accurate predictions. The main goal is to evaluate the model's performance on unseen data to ensure its reliability and effectiveness in real-world spam filtering scenarios.

Approaches and Baselines

To tackle the spam classification problem, I used a supervised machine learning approach using logistic regression as the core algorithm. To further compare the accuracy of the chosen approach, I used two additional machine learning algorithms: Naive Bayes and Random Forest. Additionally, I implemented baseline models to evaluate the performance of our approach and set benchmarks.

Approach:

Logistic Regression

Logistic regression was chosen for its simplicity and efficiency in binary classification problems. It predicts the probability of an email belonging to the spam class (label 1) or the non-spam class (label 0).

Multinomial Naive Bayes:

Naive Bayes is a simple model that uses probability to classify things. It assumes that all features (like words in text) are independent. It's often used for text classification, like sorting emails into spam or not spam, because it's quick and works well. It's also helpful to compare it to logistic regression since it gives a clear, probability-based decision.

Random Forest Classifier:

Random Forest is a more advanced model that builds lots of decision trees and combines their results to make better predictions. Random Forest can spot more complicated patterns in data compared to simpler models like Naive Bayes or logistic regression, which sometimes make stronger assumptions.

Hyperparameters

1. **Learning Rate (lr):**
    - Controls the step size of gradient updates during training.
    - Values tested: {0.1, 0.01, 0.001}.
2. **Regularization Parameter (reg_param):**
    - Penalizes large coefficients to avoid overfitting.
    - Values tested: {0, 0.01, 0.1, 1}.

Hyperparameter Tuning

I performed a grid search over the learning rates and regularization parameters using the validation set. The optimal hyperparameters were selected based on the highest validation accuracy.

Baselines

1. Trivial Baseline: Always Predict Non-Spam

- This baseline assumes that every email is non-spam (label 0).
- **Accuracy Calculation:** Accuracy = Number of Non-Spam Emails / Total Number of Emails = 3997/5729≈69.8%
- This baseline reflects the accuracy achievable without considering the input features, serving as a lower bound for model performance.

2. Random Guess Baseline

- This baseline predicts randomly between spam (1) and non-spam (0) with equal probability.
- **Expected Accuracy:** 1/2 or 50%.
- It represents a model with no learning and is another lower-bound benchmark.

Evaluation Strategy

All approaches, including baselines, were evaluated on:

- **Validation Accuracy:** Used for selecting the best hyperparameters.
- **Test Accuracy:** Assesses generalization to unseen data.

These strategies ensured that the chosen model was both efficient and reliable while outperforming naive baselines.

Measure of Success

The primary evaluation metric for this spam classification task is **accuracy**, which measures the proportion of correctly classified emails (both spam and non-spam) out of the total emails. Accuracy is computed as: test_accuracy = np.mean(y_test_pred == y_test)

Real Goal

The real goal of the task is to maximize the reliability of spam detection while minimizing false positives (non-spam emails incorrectly classified as spam) and false negatives (spam emails incorrectly classified as non-spam).

Why is Accuracy Reasonable?

Despite its limitations, accuracy remains a reasonable starting point for this project because:

1. The dataset's class imbalance is moderate (30% spam vs. 70% non-spam), reducing the risk of accuracy being overly biased.
2. As a simple metric, it is effective for comparing the model's performance against baseline methods (e.g., trivial and random guess baselines) and additional models.

Results

Learning Rate	Regularization Parameter	Validation Accuracy
0.1	0	0.99185099
	0.01	0.99185099
	0.1	0.99185099
	1	0.99185099
0.01	0	0.989522701
	0.01	0.989522701
	0.1	0.989522701
	1	0.989522701
0.001	0	0.979045402
	0.01	0.979045402
	0.1	0.979045402
	1	0.979045402

Best Hyperparameters: lr=0.1, reg_param=0

Best Validation Accuracy: 0.9918509895227008

Test Accuracy: 0.9930232558139535

Trivial Baseline Test Accuracy: 0.7825581395348837

Random Guess Accuracy (1/k): 0.5

Multinomial Naive Bayes:

Validation Accuracy: 0.9953434225844005

Test Accuracy: 0.9918604651162791

Random Forest Classifier:

Validation Accuracy: 0.9685681024447031

Test Accuracy: 0.9651162790697675

Comparison with Baselines and Additional Models

1. Trivial Baseline (Always Predict Non-Spam):

Accuracy: 78.26%

Model Test Accuracy: 99.30% (21.04% higher than the trivial baseline).

Interpretation:

The trivial baseline assumes that all emails are non-spam, which performs moderately well because non-spam emails constitute the majority class (around 70%). However, this baseline fails to detect any spam emails, which is critical in real-world spam filtering systems. The logistic regression model significantly outperforms this baseline, demonstrating its ability to effectively identify both spam and non-spam emails.

2. Random Guess Baseline:

Accuracy: 50%

Model Test Accuracy: 99.30% (49.30% higher than random guessing).

Interpretation:

The random guessing baseline assumes no learning and predicts each email as spam or non-spam with equal probability. While this sets an absolute lower bound, the logistic regression model demonstrates its capability to extract meaningful patterns from the data by getting nearly double the baseline's performance.

3. **Multinomial Naive Bayes:**

Test Accuracy: 99.19%

Interpretation:

Multinomial Naive Bayes, a common model for text classification tasks, closely rivals logistic regression. Its performance is slightly lower than that of the logistic regression model on the test set but still provides outstanding accuracy and robust classification results.

4. Random Forest Classifier:

- Test Accuracy: 96.51%

Interpretation:

The Random Forest classifier, while capturing more complex patterns through an ensemble of decision trees, achieves slightly lower accuracy than logistic regression and Naive Bayes. Despite this, it still surpasses both trivial and random baselines by a substantial margin, indicating that it successfully learns useful discriminative features from the dataset.

Related problem

There was a problem of no difference between regularization parameters during the project. I have tried making the regularization parameter wider, from 0.0001 to 10, but it outputs the same validation accuracy. This may be caused by strong features in TF-IDF representation. The features generated using TF-IDF are likely highly informative, allowing the model to generalize well across the validation set, regardless of the level of regularization. Regularization is less impactful when the features already separate the classes effectively.

Reference

Kaggle. (n.d.). *Spam email Dataset* [Dataset]. https://www.kaggle.com/datasets/jackksoncsie/spam-email-dataset/data