**SMS Spam Detection Project**

Team-Guardians Of Text

Title-Silent Gatekeepers

**Name: Dhaarna Goyal(E23CSEU0106)**

**Hiyaa (E23CSEU0120)**

**Ujjwal Saini(E23CSEU102)**

Abstract

* Brief Summary: This project explores an SMS spam detection model using machine learning to distinguish spam from non-spam (ham) messages. Leveraging a dataset of SMS messages, the project employs various pre-processing, feature extraction, and machine learning techniques to optimize spam classification accuracy.
* Problem Statement: Spam messages in SMS communications pose security and privacy concerns for users. The goal is to automate the detection of these messages using machine learning to enable efficient and reliable filtering.
* Key Findings: Through experimentation with multiple machine learning algorithms—including Naive Bayes, Support Vector Machine, and Random Forest—the Random Forest classifier provided the highest accuracy, precision, and recall scores in detecting spam messages.
* Methodology: The process involved data cleaning, transforming SMS text data into a numeric format, applying machine learning algorithms, and evaluating models using metrics like accuracy, precision, recall, and F1 score.

Introduction

* Background: SMS spam has become prevalent, affecting users' productivity and privacy. Traditional filtering methods often lack the adaptability and efficiency required for SMS content, where messages are short and often informal.
* Importance of Spam Detection: Automatic spam detection enhances user experience by eliminating unwanted messages and potentially safeguarding against fraudulent activities.
* Challenges: SMS data presents challenges due to message brevity, slang usage, and frequent abbreviations. Another challenge is imbalanced datasets, as the number of spam messages is often much lower than ham messages.

Objectives

* Primary Objective: To develop a robust machine learning model capable of classifying SMS messages as spam or ham with high accuracy.
* Specific Objectives:
  + To compare the performance of multiple machine learning algorithms on SMS data.
  + To implement data pre-processing tailored to short-text data.
  + To optimize model performance using various evaluation metrics.

Methodology

Tools and Technologies Used

* Python: Used for scripting and data analysis.
* Jupyter Notebook/GoogleCollab: For code execution, visualizations, and documentation.
* Libraries: Specific libraries for data manipulation, model building, and visualization.

Libraries Used

* Pandas: For data manipulation and storage.
* NumPy: For numerical operations.
* NLTK (Natural Language Toolkit): For text cleaning and tokenization.
* Scikit-learn: For implementing machine learning models and metrics.
* Matplotlib & Seaborn: For data visualization.

Project Design

* Data Pre-processing: The SMS data is cleaned by removing unnecessary characters, tokenizing, and stemming words.
* Feature Extraction: TF-IDF vectorization is applied to transform text into numerical data suitable for machine learning models.
* Model Selection: Multiple machine learning algorithms are tested, and model performance is evaluated based on accuracy, precision, recall, and F1 score.
* Evaluation: A comparative analysis is performed to identify the best-performing model.

Implementation Details

Data Collection and Pre-processing

* Dataset: SMS Spam Collection dataset, containing labeled SMS messages as spam or ham.
* Pre-processing Steps:
  + Text Cleaning: Removing punctuation, special characters, and extra whitespace.
  + Tokenization: Splitting text into individual words.
  + Stop Word Removal: Eliminating commonly used words that do not contribute to meaning.
  + Stemming: Reducing words to their base forms to minimize dimensionality.
* Feature Extraction: Term Frequency-Inverse Document Frequency (TF-IDF) vectorization converts text into numerical data.

Exploratory Data Analysis (EDA)

* Class Distribution Analysis: Visualizing the proportion of spam and ham messages to evaluate class imbalance.
* Word Frequency Analysis: Creating word clouds for common spam and ham terms.
* Message Length Analysis: Visualizing and comparing message lengths between spam and ham messages.

Model Creation and Testing

* Algorithms Used:
  + Naive Bayes: Known for efficiency in text classification, especially useful in handling probabilities based on word occurrence.
  + Support Vector Machine (SVM): Effective for high-dimensional data and capable of creating decision boundaries.
  + Random Forest: Robust and offers feature importance insights by aggregating results from multiple decision trees.
* Training and Testing:
  + Train-Test Split: The dataset is divided into training and test sets for accurate evaluation.
  + Cross-Validation: k-fold cross-validation ensures that the model generalizes well to new data.
* Hyperparameter Tuning: Grid search is used to optimize key parameters for each model, improving accuracy and performance.

Related Work

The paper A Review on Mobile SMS Spam Filtering Techniques by Shafi’I Muhammad Abdulhamid et al. examines the issue of SMS spam, which is an unwanted marketing practice disrupting mobile subscribers and potentially affecting service provider retention. The authors review and analyze existing spam detection, filtering, and mitigation methods, comparing popular techniques, data sets, findings, and limitations in the literature. Additionally, the paper outlines current challenges and future research directions, aiming to guide researchers toward areas requiring improvement in SMS spam detection. This review offers a comprehensive look at spam filtering methods to support advancements in mobile spam management.

The paper SMS Spam Filtering: Methods and Data by Sarah Jane Delany, Mark Buckley, and Derek Greene addresses the challenges of filtering SMS spam, a growing issue due to inexpensive bulk SMS options and the trusted nature of text messaging. While SMS spam filtering shares similarities with email spam detection, it presents unique challenges. The authors review recent developments in SMS spam filtering methods and discuss the limitations surrounding data collection and availability, which restrict research progress. They also analyze a large SMS spam dataset and present initial benchmark results, offering insights for advancing spam filtering techniques in mobile messaging.

Support Vector Machines (SVM)

Author- S.V.M. Vishwanathan, Department of Computer Science and Automation, Indian Institute of Science, Bangalore, India. M. Narasimha Murty, Department of Computer Science and Automation, Indian Institute of Science, Bangalore, India

Publisher- Published in: Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No.02CH37290)

Brief- This paper presents a novel, iterative algorithm for efficiently identifying support vectors, which are key data points defining the decision boundary in Support Vector Machines (SVM). The algorithm begins with a candidate support vector set, selected using a greedy approach to prioritize points likely to be support vectors. When adding a new point conflicts with the existing set, a backtracking technique is applied to remove obstructive points, refining the candidate set. This backtracking ensures that only essential points remain in the candidate set, which helps accelerate convergence toward an optimal set of support vectors.

To enhance the speed of convergence, the algorithm is initialized with the nearest pair of points from opposing classes, offering a solid starting boundary between the classes. From this initial boundary, the algorithm iteratively adjusts the candidate support vector set using an optimization-based method. It makes multiple passes over the dataset, pruning and adding points as needed, to ensure compliance with the Karush-Kuhn-Tucker (KKT) conditions, which are necessary for achieving the optimal decision boundary in SVM.

The algorithm’s memory efficiency is also notable, with requirements scaling as O(∣S∣1/2)O(|S|^{1/2})O(∣S∣1/2) on average, where ∣S∣|S|∣S∣ is the number of support vectors. This scaling makes it suitable for large datasets, as it can handle substantial data without excessive memory demands.

Performance evaluations on diverse real-world datasets reveal that this algorithm is highly competitive, often surpassing conventional iterative approaches like Sequential Minimal Optimization (SMO) and the Nearest Point Algorithm (NPA). These results validate its effectiveness in terms of both speed and accuracy, highlighting its potential for practical applications in data-intensive fields.

Python’s Toolbox: A Comprehensive study of Python Library Tools for Empowering Development

Author- Leon John Jaison 2 nd Integrated MSc AI and ML Department of Computer Science, Ninitha Mary Antonina N.M Faculty Department of Computer Science

Publisher- Department of Computer Science Bharata Mata College Thrikkakara

Brief- This paper delves into Python's vast ecosystem of tools and libraries, showcasing their impact on software development productivity and quality. Python’s popularity stems largely from its versatility and ease of integration across various domains, which has made it an essential part of many modern software development workflows. The extensive range of libraries and tools available within Python enables developers to streamline complex tasks, boost productivity, and produce cleaner, more efficient code.

Core libraries like NumPy and Pandas are central to Python’s utility in data science and analytics, offering powerful capabilities for data manipulation, processing, and analysis. NumPy allows for efficient computation of large, multi-dimensional arrays, while Pandas offers data structures and functions designed to work with structured data. These libraries have become indispensable for data scientists, simplifying tasks that would otherwise require extensive coding.

Beyond data science, Python provides specialized libraries for more specific needs. For instance, Requests is a popular library for handling HTTP requests, enabling straightforward communication with web servers and APIs. Flask, a lightweight web framework, allows developers to build web applications with minimal setup, making it ideal for projects where flexibility and quick deployment are essential. Additionally, Python’s tools extend to fields like artificial intelligence (AI) and machine learning (ML). Libraries such as scikit-learn and TensorFlow provide high-level interfaces for building and deploying machine learning models, making Python a dominant language in AI research and application development.

Python’s ecosystem also includes libraries for data visualization, testing, automation, and more, each contributing to a streamlined development experience.

LOGISTIC REGRESSION

Author- Wright, R. E. (1995). Logistic regression. In L. G. Grimm & P. R. Yarnold (Eds.), Reading and understanding multivariate statistics (pp. 217–244)

Publisher- American Psychological Association

Brief- Logistic regression is a statistical method used to model the relationship between a set of predictor variables and a dichotomous (binary) or polytomous (multinomial) outcome variable. Unlike linear regression, which predicts a continuous outcome, logistic regression predicts the probability of a categorical outcome, typically by transforming it using a logistic (sigmoid) function. This approach can be extended to handle more than two categories of the outcome variable, known as multinomial logistic regression.

In terms of similarities, both logistic and linear regression aim to model relationships between predictor and outcome variables and share methods for hypothesis testing and evaluating model fit. However, they differ in that linear regression predicts continuous outcomes, while logistic regression is designed for categorical data and uses log-odds for linearization. Additionally, logistic regression does not assume normality, homoscedasticity, or linearity in the relationships between variables; instead, it assumes independence of errors and a sufficiently large sample size.

To interpret logistic regression analysis, coefficients indicate the change in log-odds for a one-unit increase in the predictor variable. For example, in a hypothetical study on smoking (predictor) and lung disease (binary outcome), a positive coefficient for smoking would indicate higher odds of disease.

In studies with multiple predictors, coefficients reveal each predictor’s unique effect, controlling for others. For complex models with many predictors, stepwise selection or regularization methods help build models by selecting the most significant predictors. Results of logistic regression include odds ratios, significance tests, and classification accuracy, all essential for interpreting the model.

The domain of SMS spam detection has seen significant contributions in natural language processing and machine learning. Traditional methods like keyword matching and rule-based filtering are limited by the diversity of spam language. Machine learning approaches have shown superior performance, with models such as Naive Bayes and SVM being particularly popular. Recent studies suggest that ensemble models like Random Forest can further improve accuracy. Additionally, advancements in NLP, such as BERT and Word2Vec embeddings, hold potential for SMS spam detection in future studies.

Data Pre-processing

* Cleaning Text Data: Text was pre-processed to remove noise, special characters, and irrelevant punctuation.
* Tokenization and Stop Word Removal: The dataset was tokenized and stripped of stop words to focus on meaningful content.
* Stemming and Lemmatization: Stemming reduced words to their base forms, standardizing terms.
* TF-IDF Vectorization: This technique transformed the text into a numerical format, allowing models to assess the importance of terms within each message.

Data Visualization / EDA using Python Libraries

* Class Distribution: Visualized using bar charts to observe the balance between spam and ham messages.
* Word Cloud: Word clouds generated for spam and ham messages highlighted common terms, aiding in feature selection.
* Message Length Analysis: Compared the distribution of message lengths in spam versus ham, indicating trends in spam characteristics.
* Correlation Analysis: Used to examine correlations between word frequencies and spam probability.

Model Creation and Testing

* Naive Bayes: Provided a reliable baseline, achieving reasonable accuracy due to its probabilistic nature. However, it had limitations in capturing complex relationships in data.
* Support Vector Machine (SVM): Showed competitive accuracy with the ability to handle high-dimensional data, though it required tuning for optimal results.
* Random Forest: Delivered the best overall performance, achieving the highest F1 score. Its ensemble nature provided stability in predictions and insights into feature importance.
* Evaluation Metrics:
  + Accuracy: Overall effectiveness of the model in predicting spam versus ham messages.
  + Precision and Recall: Precision captured the model's accuracy in identifying true spam messages, while recall measured its ability to find all spam messages.
  + F1 Score: Provided a balanced measure of the model’s accuracy and reliability.
* Best Model: Random Forest achieved an accuracy of over 90%, along with high precision and recall. It outperformed other models, making it the most reliable choice for this application.

Conclusion

* Summary: The project demonstrated the feasibility of using machine learning to classify SMS messages as spam or ham. With appropriate pre-processing and feature engineering, models like Random Forest can deliver robust performance.
* Key Findings: Random Forest achieved the highest accuracy, precision, and recall. The importance of thorough data pre-processing was evident in improving model effectiveness.
* Future Work: Future improvements could involve integrating advanced NLP methods, such as BERT embeddings, to further enhance model performance. Deploying the model as an SMS filtering tool could be an interesting application.

References

* ~ Kaggle - https://www.kaggle.com/
* ~ Python Libraries

- https://docs.python.org/3/library/index.html

* ~ W3 Schools

https://www.w3schools.com/python/python\_ml\_train\_test.asp