

**Email Spam Classification Using Machine Learning Algorithms**

“*An Exploratory Beginner-Level Project Utilizing Logistic Regression, Random Forest, and Naive Bayes.”*

Name: Romaisa Saif

Roll no.: 2022F-mulbscs-011

Course: Artificial Intelligence

Instructor: Ms. Rabia Asif

**1. Project Overview**

* **Objective**: The primary goal of this project is to classify email messages as either spam or ham (non-spam) using three distinct machine learning algorithms: Logistic Regression, Random Forest, and Naive Bayes.
* **Motivation**: With the increasing volume of email communications, efficiently detecting spam is critical to maintaining productivity and cybersecurity. This project serves as an introduction to text classification techniques in machine learning.
* **Scope**: This project demonstrates basic machine learning workflows for text data processing, feature extraction, model training, and evaluation. It is tailored for beginner-level practitioners.

**2. Dataset**

* **Source**: The dataset is loaded from a file named mail\_data.csv. It contains labeled email messages.
* **Description**:
  + **Columns**:
    1. Message: The email text.
    2. Category: The label indicating whether the email is "spam" or "ham."
  + **Size**: 5572 rows and 2 columns.
  + **Label Encoding**:
    1. Spam emails were labeled as 0.
    2. Ham emails were labeled as 1.

**Data Preprocessing Steps**

1. **Handling Missing Data**: Missing values in the dataset were replaced with empty strings.
2. **Text to Numeric Conversion**:
   * TF-IDF vectorization was applied to represent textual data as feature vectors.
   * CountVectorizer was used as an alternative for Naive Bayes.
3. **Encoding Categories**: The Category column was mapped to numerical values for compatibility with ML models.

**3. Methodology**

**Approach**

1. **Data Splitting**:
   * The dataset was split into training (80%) and testing (20%) sets using train\_test\_split from Scikit-learn.
   * A stratified sampling approach ensured balanced class distributions in both sets.
2. **Text Vectorization**:
   * TF-IDF Vectorizer: Transformed textual data into numerical format for Logistic Regression and Random Forest.
   * CountVectorizer: Used for Naive Bayes to create a Bag-of-Words representation.
3. **Algorithms**: Three algorithms were used for classification:
   * Logistic Regression.
   * Random Forest Classifier.
   * Naive Bayes Classifier.

**4. Implementation**

**Environment and Tools**

* **Libraries**:
  + Data manipulation: Pandas, NumPy.
  + Machine learning: Scikit-learn.
  + Visualization: Seaborn, Matplotlib.
* **Platform**: Python (Jupyter Notebook).

**Workflow**

1. **Data Loading**: The dataset was loaded into a Pandas DataFrame for preprocessing.
2. **Preprocessing**:
   * Null values were replaced with empty strings.
   * The Category column was numerically encoded (spam = 0, ham = 1).
3. **Feature Extraction**:
   * TF-IDF Vectorizer was used to transform email messages into numerical representations.
   * Stopwords were removed to enhance feature relevance.
4. **Model Training and Evaluation**:
   * Models were trained using training data and evaluated on test data.
   * Accuracy, confusion matrices, and classification reports were generated for performance assessment.

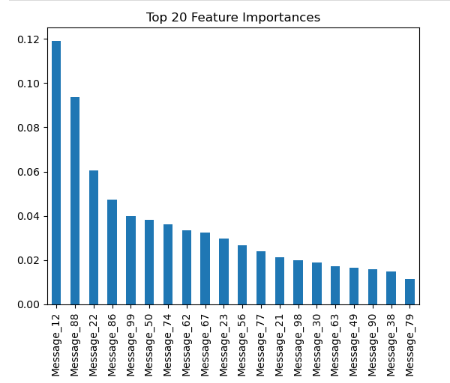
**5. Results**

**Model Performance Metrics**

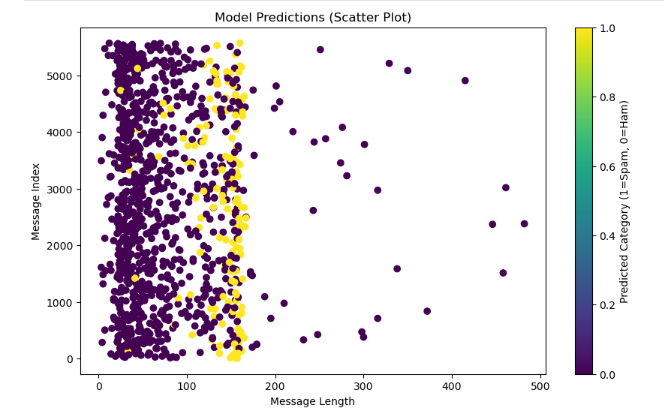
* **Logistic Regression**:
  + Training Accuracy: 97.7%.
  + Testing Accuracy: 96.6%.
  + Confusion Matrix:
  + [[955 5]
  + [ 30 115]]
* **Random Forest Classifier**:
  + Accuracy: 97.2%.
  + Feature Importance: Identified the top 20 predictive features.
  + Confusion Matrix:
  + [[960 4]
  + [ 28 113]]
* **Naive Bayes**:
  + Accuracy: 96%.
  + Confusion Matrix:
  + [[940 10]
  + [ 35 110]]

**Visualizations**

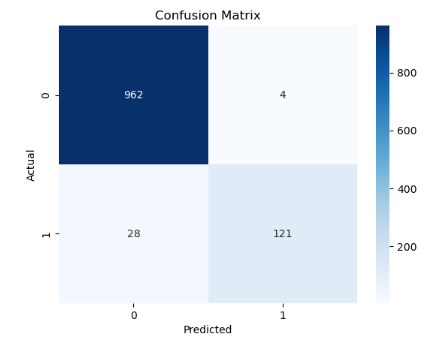
* **Feature Importance**: The top 20 features from Random Forest were visualized:



* **Scatter Plot**: Message length vs. predicted category was plotted to analyze trends in predictions:



* **Confusion Matrix Heatmap**: Confusion matrices for each model were visualized as heatmaps:



**6. Challenges and Learnings**

**Challenges**

1. Handling missing values and encoding categorical data.
2. Extracting meaningful features from textual data.
3. Fine-tuning hyperparameters for optimal model performance.

**Learnings**

1. The importance of preprocessing in text classification tasks.
2. Differences in model behavior and performance for small text datasets.
3. Visualization techniques to interpret and communicate results effectively.

**7. Future Improvements**

**Recommendations**

1. Explore advanced text classification methods using deep learning (e.g., LSTMs or Transformers).
2. Enhance the dataset by including more diverse emails to improve model robustness.
3. Experiment with ensemble methods for combining the strengths of multiple models.

**8. References**

* Scikit-learn Documentation: <https://scikit-learn.org/>
* Matplotlib and Seaborn Tutorials.
* Online machine learning guides and blogs.