# **Email Spam Classifier Project Report**

## **1. Introduction**

This project centers on developing an **Email Spam Classifier** capable of accurately distinguishing spam from non-spam emails. By leveraging multiple datasets sourced from **Kaggle** and applying various preprocessing, feature engineering, and machine learning techniques, this project aims to create a robust classification model. The final classifier achieves high precision and accuracy, with the **Random Forest Classifier** emerging as the best-performing model.

## **2. Project Goals**

* Develop a reliable and accurate spam classifier.
* Preprocess and standardize text data, removing irrelevant content.
* Engineer features to improve the model’s performance and interpretability.
* Address data imbalance issues common in spam classification.
* Evaluate and fine-tune multiple machine learning models for optimal results.

## **3. Methodology**

### **3.1 Data Preprocessing**

Data preprocessing is a critical part of this project, aimed at creating a clean, standardized dataset for model training. The following steps were performed:

1. **Lowercase Conversion**: Ensures uniformity by converting text to lowercase.
2. **Whitespace and Newline Removal**: Removes extra spaces and newline characters to make the text more compact.
3. **Stopwords Removal**: Eliminates common stopwords, while retaining essential ones like "not" and "but" to preserve context.
4. **Lemmatization**: Reduces words to their base form, aiding in consistency across variations of the same word.
5. **Text Cleaning**: Strips out URLs, email addresses, HTML tags, and non-English characters.

### **3.2 Feature Engineering**

To enhance the model’s learning capacity, several features were engineered:

* **Total Word Count**: Computed before and after preprocessing to capture the effect of cleaning.
* **Character Count**: Measures the number of characters both before and after preprocessing.
* **Stopwords Count**: Quantifies stopwords in the text, providing insight into language patterns.
* **Punctuation Count**: Counts punctuation marks as an indicator of email structure and tone.

### **3.3 Contextual Embedding**

The **Word2Vec** model was employed to generate contextual embeddings for the tokenized text. A 100-dimensional vector represents each email, capturing semantic relationships among words. This approach enables the model to leverage the contextual meaning of words, improving classification performance.

## **4. Model Development and Evaluation**

### **4.1 Handling Imbalanced Data**

Since the spam dataset exhibited class imbalance, various techniques were applied to ensure balanced data:

* **Undersampling**: Reduced the majority class size to achieve a more balanced class distribution.
* **Oversampling**: Duplicated instances from the minority class.
* **SMOTE (Synthetic Minority Over-sampling Technique)**: Generated synthetic examples for the minority class to further enhance balance.

### **4.2 Model Training**

Several models were trained to determine the best classifier for this project:

* **Logistic Regression**: Provided a strong baseline with good accuracy.
* **Random Forest Classifier**: Showed the best performance across precision, recall, and F1-score.
* **XGBoost Classifier**: Delivered competitive results but required extensive tuning.
* **Decision Tree Classifier**: Served as a simpler alternative but was outperformed by ensemble methods.

### **4.3 Hyperparameter Tuning for Optimal Performance**

The **Random Forest Classifier** was selected as the primary model and fine-tuned with these hyperparameters for optimal results:

* **n\_estimators**: 100
* **max\_depth**: 20
* **min\_samples\_split**: 5
* **min\_samples\_leaf**: 2
* **max\_features**: 'sqrt'
* **class\_weight**: 'balanced'

Following tuning, the Random Forest model consistently achieved precision and accuracy rates above **0.90**, making it the ideal classifier for this task.

## **5. Results**

The **Random Forest Classifier** proved to be the best-performing model. Here are the results summary:

* **Random Forest Classifier**: Achieved the highest precision, accuracy, and F1-score.
* **Logistic Regression**: Performed effectively after balancing the data, reaching around **0.90** in both precision and accuracy.

The high scores across evaluation metrics confirm that the classifier meets the project’s goal of effective spam detection.

## **6. Challenges and Solutions**

### **6.1 Imbalanced Dataset**

Spam classification often suffers from imbalanced datasets, where the non-spam class outnumbers the spam class. By applying undersampling, oversampling, and SMOTE, this imbalance was addressed, leading to better model performance.

### **6.2 Model Selection and Performance**

Finding the optimal model required training multiple algorithms and extensive tuning. The **Random Forest Classifier** emerged as the best choice after comparison with alternatives, delivering superior results with the fewest errors.

## **7. Future Work**

This project lays the groundwork for a highly accurate spam classifier, yet several avenues remain for further exploration:

* **Advanced Models**: Implementing deep learning models like **LSTM** or **Transformer-based models** could capture even more nuanced patterns in email text.
* **Enhanced Embeddings**: Utilizing pre-trained models such as **BERT** may improve contextual understanding.
* **Data Augmentation**: Experimenting with additional augmentation techniques could further address class imbalance and improve model robustness.

## **8. Conclusion**

This Email Spam Classifier project demonstrates an effective approach to identifying spam emails through rigorous preprocessing, feature engineering, and handling of imbalanced data. The **Random Forest Classifier**, after hyperparameter tuning, stands out as the best model, achieving high precision and accuracy. Future work on more advanced models and embeddings could continue to improve upon this already effective solution.