# SULTAN 56189

# CH. MUBSHIR 56892

# AI PROJECT'S DOCUMENTATION

# Text Classification with Python: Build and Compare Three Text Classifiers

## Table of Contents



---

## 1. Introduction

In this project, we will explore text classification using Python. We will build and compare three different text classifiers: Multinomial Naive Bayes, Complement Naive Bayes, and Support Vector Classifier. The objective is to classify text messages as either spam or ham (non-spam) using a dataset sourced from Kaggle.

## 2. What is Text Classification?

Text classification is the process of assigning a label to a piece of text based on its content or context. Common applications include:

- **Spam Detection**: Identifying spam emails or messages.
- **Sentiment Analysis**: Classifying text as positive, negative, or neutral.
- **Document Categorization**: Organizing documents into predefined categories such as business, politics, etc.

Text classification typically employs supervised learning, where a labeled dataset is used to train the model.

## 3. Dataset Preparation

## 3.1 Sourcing the Dataset

The dataset for this project is obtained from Kaggle, specifically designed for spam message classification. It contains two columns:

- **Message**: The text of the message.
- **Category**: The label indicating whether the message is spam or ham.

## 3.2 Exploring the Dataset

Before proceeding, it is essential to explore the dataset to understand its structure and the distribution of categories.

# 4. Data Preprocessing

## 4.1 Cleaning the Data

Data preprocessing is crucial for effective model training. Key steps include:

- **Removing Stop Words**: Words that do not contribute to the classification are removed.
- **Converting Text to Numerical Format**: Text data is transformed into a numerical format using TF-IDF (Term Frequency-Inverse Document Frequency).

## 4.2 TF-IDF Vectorization

TF-IDF assigns weights to words based on their frequency in a document relative to their frequency across all documents. This helps in focusing on words that are significant for classification.

# 5. Model Training

## 5.1 Splitting the Dataset

The dataset is split into training and testing sets. Typically, 70-80% of the data is used for training, while the remaining 20-30% is reserved for testing the model's performance.

## 5.2 Creating Pipelines for Classifiers

Pipelines are created for each classifier to streamline the process of transforming the data and fitting the model. The classifiers used are:

- Multinomial Naive Bayes
- Complement Naive Bayes
- Support Vector Classifier

# 6. Model Evaluation

## 6.1 Making Predictions

After training, the models are used to make predictions on the test set.

## 6.2 Evaluating Performance

The models are evaluated using accuracy scores and classification reports, which provide insights into precision, recall, and F1-score.

# 7. Results

The results indicate that the Support Vector Classifier achieved the highest accuracy, followed by Complement Naive Bayes and Multinomial Naive Bayes. The performance metrics highlight the effectiveness of each classifier in handling the spam detection task.

# 8. Conclusion

In this project, we successfully built and compared three text classifiers for spam detection. The Support Vector Classifier outperformed the others, demonstrating its effectiveness in text classification tasks. Future work could involve experimenting with additional preprocessing techniques, hyperparameter tuning, or exploring other classification algorithms.