**Spam Detection**

**Introduction**

Spam messages are unsolicited messages that can be a nuisance and, in some cases, harmful. Identifying spam messages using machine learning is an essential application of natural language processing (NLP). In this session, we will explore how to classify SMS messages as spam or ham (legitimate) using Python and machine learning techniques.

**Key Concepts**

To understand spam detection, we need to cover the following concepts:

**1. Natural Language Processing (NLP)**

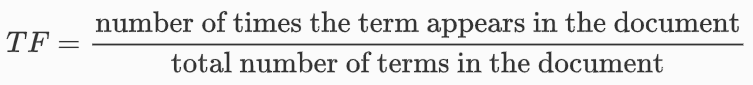
NLP is a field of artificial intelligence that helps computers understand, interpret, and manipulate human language. Spam detection is a text classification problem within NLP.

**2. TF-IDF (Term Frequency - Inverse Document Frequency)**

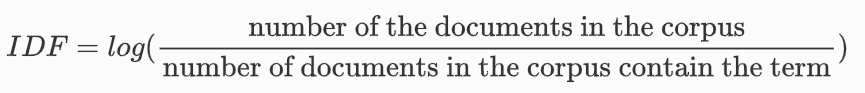
TF-IDF is a statistical measure used in natural language processing and information retrieval to evaluate the importance of a word within a document relative to a collection of documents, known as a corpus. It helps identify how significant a word is in a particular document by balancing its frequency in that document against its prevalence across the entire corpus.

**Components of TF-IDF:**

1. **Term Frequency (TF):** This component measures how often a term appears in a document. A common approach is to divide the number of times the term appears by the total number of terms in the document, resulting in a normalized value that accounts for document length.



1. **Inverse Document Frequency (IDF):** This component assesses the rarity or commonality of a term across all documents in the corpus. It is calculated by taking the logarithm of the ratio of the total number of documents to the number of documents containing the term. Terms that appear in many documents receive a lower IDF score, indicating they are less significant, while terms found in fewer documents receive a higher IDF score, highlighting their importance.



The TF-IDF value is obtained by multiplying the TF and IDF scores for a term. A high TF-IDF score indicates that a term is relatively frequent in a specific document but rare across the entire corpus, suggesting it carries significant information about that document's content.

A black text on a white background

AI-generated content may be incorrect.

**3. Text Preprocessing**

Before training a machine learning model, text data needs to be cleaned and prepared. This involves:

* Removing special characters
* Converting text to lowercase
* Removing stopwords (e.g., "the," "is")
* Tokenization (splitting text into words)
* Stemming/Lemmatization (reducing words to their root forms)

**4. Classification Models**

We use classification models to differentiate between spam and ham messages. Common models include:

* **Naïve Bayes:** Often used for text classification due to its simplicity and effectiveness.
* **Logistic Regression:** Suitable for binary classification problems.
* **Support Vector Machines (SVM):** Works well for high-dimensional spaces like text data.

**Dataset Overview**

The dataset contains SMS messages labeled as spam or ham. Each row consists of:

* **Label:** "spam" or "ham"
* **Message:** The actual SMS content

**Python Code Explanation**

The provided Python script follows these steps:

1. **Load the Dataset:**
2. import pandas as pd
3. df = pd.read\_csv("spam.csv", encoding='latin-1')[['v1', 'v2']]
4. df.columns = ['label', 'message']

This loads the dataset and renames the columns for better readability.

1. **Preprocessing:**
2. from sklearn.model\_selection import train\_test\_split
3. from sklearn.feature\_extraction.text import TfidfVectorizer
4. from sklearn.naive\_bayes import MultinomialNB
5. from sklearn.pipeline import Pipeline
   * Splitting data into training and test sets
   * Using TF-IDF for feature extraction
   * Applying Naïve Bayes for classification
6. **Training the Model:**
7. model = Pipeline([
8. ('tfidf', TfidfVectorizer()),
9. ('classifier', MultinomialNB())
10. ])
11. model.fit(X\_train, y\_train)

The pipeline vectorizes the text and trains the Naïve Bayes classifier.

1. **Evaluating the Model:**
2. from sklearn.metrics import accuracy\_score, classification\_report
3. y\_pred = model.predict(X\_test)
4. print("Accuracy:", accuracy\_score(y\_test, y\_pred))
5. print(classification\_report(y\_test, y\_pred))

This prints accuracy and other performance metrics.

**Practical Implementation**

* Install necessary libraries: pip install pandas scikit-learn
* Run the script and observe results
* Experiment with different classifiers (e.g., SVM, logistic regression)
* Try additional preprocessing techniques (e.g., removing stopwords)

**Conclusion**

Spam detection is a vital application of machine learning. Understanding text processing, TF-IDF, and classification models allows us to build efficient spam filters. This session provides a hands-on approach to implementing spam detection using Python.