# ML Lab 10

## What was the accuracy for each of the following settings?

|  |  |  |
| --- | --- | --- |
| **Vectorization Method** | **Preprocessing** | **Accuracy** |
| TF-IDF | No | 0.9576 (95.76%) |
| TF-IDF | Yes | 0.9641 (96.41%) |
| CountVectorizer | No | 0.9763 (97.63%) |
| CountVectorizer | Yes | 0.9763 (97.63%) |

## Which method achieved the highest accuracy? Why do you think that is?

Highest Accuracy - CountVectorizer with and without preprocessing at 97.63%.

* Simplicity - CountVectorizer captures raw frequency, which may be more effective when distinguishing between spam and ham since spam often contains repeated phrases like "free", "win", or "call now".
* TF-IDF Downweights Common Terms - TF-IDF penalizes words that are common across documents, even if those are key spam indicators.
* Spam is Pattern-Based - In spam detection, exact word occurrences and patterns matter more than uniqueness, which CountVectorizer captures better.
* Role of Preprocessing - Preprocessing helps slightly with TF-IDF by reducing noise and standardizing tokens. But for CountVectorizer, preprocessing doesn’t change accuracy.

## Which method performed the worst, and why might that be?

Worst Performing - TF-IDF without preprocessing with 95.76% accuracy.

* Noise and Redundancy - Without preprocessing, the text may include unnecessary symbols, mixed casing, or common stopwords that dilute meaningful signal.
* No Normalization - Variants like “FREE”, “Free”, and “free” are treated differently, reducing model generalization.
* Useful tokens missed - Stemming and stopword removal are essential for improving representation in sparse TF-IDF space; without them, rare word forms may not contribute effectively.