Notebook Flow Description

1. Data Cleansing:

1.1 Text cleaning: Operations like removing special characters, HTML tags, or email addresses are performed as the first step since they eliminate irrelevant information from the raw text. It ensures that the next processing steps, like tokenization, operate on a more refined input.

1.2 Tokenization: We use it as the second step after text cleaning because tokenization breaks down the cleaned text into individual words or tokens, making it ready for further processing.

1.3 StopWord Removal: This step is performed after tokenization because it operates on individual tokens and removes those identified as stopwords.

1.4 Lemmatization: This technique is applied after tokenization because it operates on individual tokens rather than entire texts and after stopWord removal to reduce computational overhead and ensures that the remaining tokens are processed more accurately.

2. Data Splitting: Data splitting after data cleansing ensures that both the training and testing datasets maintain the same level of cleanliness, which guarantees an unbiased evaluation of the model's performance, as it is trained on clean data and tested on similarly cleaned, unseen data.

3. Text Embedding: We use text embedding on training set after splitting the data to ensure fairness in evaluating the model. This way, the numerical representations of the text come only from the training data, so the model is tested on unseen text fairly.

4. Models Training: We train the models after text embedding because embedding converts textual data into a numerical format that models can understand. This allows the models to learn patterns from the numerical representations and make classification based on them.

5. Models Evaluation: Model evaluation comes after model training to assess how well the trained model performs on unseen data. This step ensures that the model's performance is assessed before deployment, helping identify any issues and improve its accuracy and reliability.

Data Preprocessing & Features Extraction

1. Text Cleaning: This involves removing noise, such as special characters or HTML tags, from the text data, which ensures that the text is standardized and free from irrelevant information.

2. Tokenization: Tokenization breaks down the text into individual words or tokens. This process is essential as it provides the model with the basic units of text to analyze and classify.

3. StopWord Removal: Removing stopwords such as "the," "is," or "and" helps reduce noise and improve the efficiency of the model by focusing on more meaningful words.

4. Lemmatization: Reducing words to their base form helps in spam filtering as it ensures that variations of words (run,running,..) treated as same word, reducing the complexity of the model.

➢ **We do not typically use stemming or POS tagging for spam filtering because:**

1. Stemming: It may result in loss of meaningful information. Spam filtering requires accurate understanding of the content, lemmatization is preferred as it produces more accurate results.

2. POS Tagging: POS tagging is not necessary for spam filtering, which focuses more on the presence or absence of specific keywords or patterns associated with spam. Therefore, it adds unnecessary complexity without significant benefit to the task.

Data Splitting:

We used a 60-40 split for training and testing data to balance having enough data to train the model effectively while ensuring a sizable test set for robust evaluation. Setting a random state ensures that the train and test sets are representative of overall dataset, avoiding bias from specific ordering. It guarantees split consistency for different code runs.

Model Training

For Classifiers:

We chose logistic regression and support vector machines (SVMs) as classifiers due to their effectiveness for text classification tasks. Both models offer strong performance and are widely used in spam filtering and other text classification applications. Logistic regression is chosen for its simplicity and interpretability, making it suitable for binary classification tasks like spam detection. SVM excels at handling complex decision boundaries and high-dimensional data.

For text embedding techniques:

➢ We use Bag of Words (BoW) and TF-IDF because they are simple, widely used, and effective.

- **BOW:** is straightforward and efficient for fundamental text analysis tasks.
- **TF-IDF:** addresses limitations of BOW by giving weight to words that are rare in the entire doc collection but common in a specific doc. This helps capture the importance of words accurately.

➢ We use Word2Vec and Doc2Vec neural network-based text embedding techniques because they effectively capture semantic relationships and contextual information in text data.

- **Word2Vec:** learns dense vector representations of words, facilitating tasks like word similarity and sentiment analysis.
- **Doc2Vec:** extends this to entire documents, enabling tasks such as document classification and similarity analysis.

Model Evaluation:

We utilize accuracy, precision, recall, and F1-score to comprehensively evaluate the spam filtering model. Accuracy measures overall correctness in predictions. Precision focuses on reducing false positives. Recall emphasizes minimizing false negatives. F1-score strikes a balance between precision and recall, offering a holistic assessment.

Dominant Models

Model	Accuracy	Precision	Recall	f1-score
Tfidf Logistic Regression	0.980595	0.995781	0.944	0.969199
Tfidf SVM	0.992238	0.997283	0.978667	0.987887
Bag of Words Logistic Regression	0.992238	0.998638	0.977333	0.987871
Bag of Words SVM	0.987063	0.994505	0.965333	0.979702
Word2Vec Logistic Regression	0.984907	0.991747	0.961333	0.976303
Word2Vec SVM	0.985339	0.991758	0.962667	0.976996
Doc2Vec Logistic Regression	0.961621	0.963534	0.916	0.939166
Doc2Vec SVM	0.959897	0.960729	0.913333	0.936432

Both Tfidf SVM and BOW Logistic Regression exhibit high precision, recall, accuracy, and F1-score. Tfidf and BOW models tend to outperform Word2Vec and Doc2Vec in classifying accuracy, particularly in spam detection tasks because it relies heavily on keyword frequency and the simplicity and interpretability of Tfidf and BOW models make them well-suited for spam filtering.