Notebook Flow Description:

1st, we **load the dataset** and perform initial data exploration by discover top 5 rows in the data (head), display data information, and count values of the target (0s and 1s) to show data balance.
2nd, we perform **date preprocessing and feature extraction** by applying text cleaning, tokenization, and lemmatization. Then we save the results again by replacing original data with the cleaned one.
3rd, we identify text embedding techniques whether neural networks based (word2vec and doc2vec) or not (bag of words and tf-idf).
4th, we perform **data splitting** randomly in the portion of 60/40 (60 for training and 40 for testing).
5th, we identify model classifiers (Logistic Regression, Decision Tree, and SVM) that will be used in **model training**.
6th, we identify **model evaluation** metrics (Accuracy, Precision, Recall, and F1-Score) that will be used to evaluate the models.
Finally, we create all models and evaluate their performance, then we display all the evaluation results of the models in a single data frame to summarize and compare them easily.
You can see more details below.

Data Preprocessing & Features Extraction:

Below are the steps we did:
1. Text Cleaning: This initial step includes removing email addresses and HTML content from the text, we also remove all punctuation marks and numbers. Eliminating these elements helps in focusing only on the meaningful text.
2. Tokenization: Here, the cleaned text is broken down into individual elements "tokens". Tokenization is done after cleaning to make sure that the tokens are useful and relevant. This step is very important to the next steps.
3. Removing Stopwords: Once the text is tokenized, we removed the stop words. Removing them allows the focus to shift to more meaningful words which could have a stronger relevance to the analysis.
4. Lemmatization: The final step includes converting different forms of a word into a single, base form (lemmatization). This process helps in grouping similar terms together, which simplifies the analysis and enhances the learning effectiveness of the models by reducing the complexity of the text.

Data Splitting:

We choose to do 60/40 split to ensure that the model learns well from the data, and this balance also ensures that the model will test on enough data. We do it through the using of 'train_test_split' function provided from sklearn library, splitting into random train and test subsets.

Model Training:

Classifiers:

1. **Logistic Regression:** We chose it because the output is categorical (spam or ham), and it can make a given input belongs to a certain class using the probability.
2. **Decision Tree:** We chose it because effectively capture complex relationships within the data without requiring extensive preprocessing, less prone to overfitting and works well with categorical data.
3. **Support Vector Machine:** We chose it because of our text data is high-dimensional (as there are large vocabulary can be extracted from it), and the support vector machine model is very effective with dealing with the high-dimensional data.

Text Embedding Techniques:

Techniques that aren’t based on neural networks:

1. **TF-IDF:** very effective to prepare our text data for the classification task, and it helps in highlighting the features that are most relevant for distinguishing between different categories of emails.
2. **Bag of Words:** it mainly looks at how often each word appears in the text. And this is very straightforward, so it is easy to handle and process large amounts of emails quickly.

Techniques that are based on neural networks:

1. **Word2Vec:** it considering the context of words, which enhances classification tasks by capturing semantic relationships. In this vector space, words with similar meanings are located near each other, so it helps identify patterns like spam in emails. We used a Continuous Bag of Words (CBOW) as suitable for large datasets with repetitive words.
2. **Doc2Vec:** it efficiently handles email classification by capturing the overall meaning of emails rather than individual words. Its computational efficiency makes it ideal for processing large email volumes. By disregarding word order, it accurately discerns spam from ham emails solely based on content, enhancing classification effectiveness.

Model Evaluation:

1. **Accuracy:** We chose to use accuracy to tells us the percentage of emails that our model correctly identifies as spam or ham. It gives us a quick idea of how well the model is working overall.
2. **Precision:** We chose to use precision to reflect the model's ability to classify spam and ham emails correctly. It does that by computing the ratio of correctly predicted positive observations to the total predicted positives.
3. **Recall:** We chose to use recall to detect the spam emails, as it indicates how many actual spam emails are detected correctly by the classifier model.
4. **F1-Score:** We chose to use the F1-Score because our data isn't evenly distributed between spam and ham emails, and this score is very useful when the class distribution is imbalanced.

Dominant Models:

All models’ performance results (table summarizing performance metrics (%)), based on testing set:

Text Embedding Technique	Classification Model	Accuracy	Precision	Recall	F1-Score
Word2Vec	Logistic Regression	98.5339	98.9071	96.5333	97.7058
Word2Vec	Decision Tree	97.1971	95.1252	96.2667	95.6925
Word2Vec	Support Vector Machine	98.4476	99.0385	96.1333	97.5643
Doc2Vec	Logistic Regression	97.1539	97.2376	93.8667	95.5224
Doc2Vec	Decision Tree	86.2009	77.9221	80.0	78.9474
Doc2Vec	Support Vector Machine	97.3696	97.7809	94.0	95.8532
TF-IDF	Logistic Regression	98.6201	99.5856	96.1333	97.829
TF-IDF	Decision Tree	97.4989	95.6464	96.6667	96.1538
TF-IDF	Support Vector Machine	99.2238	99.7283	97.8667	98.7887
Bag of Words	Logistic Regression	99.4394	99.8647	98.4	99.1269
Bag of Words	Decision Tree	98.1026	97.832	96.2667	97.043
Bag of Words	Support Vector Machine	98.7495	99.4513	96.6667	98.0392

The dominant models are SVM with TF-IDF and Logistic Regression with BOW, since they achieve the highest score across all metrics. which show the model's ability to identify correctly mails as spam or ham.