CA02 Strategy Report- Jerry and Nicholson

1. Goal of This Project

The goal is to build a model to classify emails into: Spam (1) and Not Spam (0) We use Naive Bayes to learn patterns from training emails, then test the model on testing emails and report accuracy.

2. Data Understanding

This dataset is not a CSV file. It is a folder-based dataset:

- train-mails/ = training emails
- test-mails/ = testing emails

Each file is one email (a text file).

How labels are decided

There is no label column. The label is from the **file name**:

- If the file name starts with **"spmsg"**, it is **Spam (1)**
- Otherwise, it is **Not Spam (0)**

This rule is used in both training and testing.

3. Sample Code Strategy / Logic

Step 1 — Build Word Dictionary

Read all emails → Split text into words → Remove: non-alphabet words and single-letter words → Count word frequency → Keep top 3000 most frequent words as features.

Step 2 — Feature Extraction

Each email becomes one row → Each dictionary word becomes one column → Cell value = number of times the word appears in the email → This converts text into numeric data.

Step 3 — Label Creation

Labels are created from file names → spmsg* → Spam (1) → Others → Not Spam (0)

Step 4 — Model Training

Use training feature matrix and labels → Train Naive Bayes model → Model learns word patterns for spam and non-spam.

Step 5 — Model Testing

Convert test emails using the same dictionary → Predict spam or not spam → Calculate accuracy.

4. Weakness of Sample Code

Weakness 1 — Model Choice

The sample code uses Gaussian Naive Bayes but our features are word counts, which are discrete. Gaussian NB assumes the data is continuous and follows a normal distribution. So it is not the best match for this problem. The better choice is using Multinomial Naive Bayes.

Weakness 2 — Dictionary built only from training

The slide says to use all emails (train + test) to build the dictionary. The sample code builds dictionary from training only.

Weakness 3 — Only uses one line of email text

In the sample feature extraction, it only uses a specific line (line index = 2). This may miss important information in the full email body. The Better choice is use the full email text.

Weakness 4 — Slow feature extraction

The sample code checks each word by looping over all 3000 dictionary words repeatedly, this is slow and not efficient. The better choice is to create a word-to-index map, so we can find the column quickly.

Weakness 5 — File path splitting may break on Windows

The sample code splits file path using /. On Windows it may not work correctly

because Windows uses \. Th better choice is to use os.path.basename() to safely get file name.

5. Improved Design Plan

- Use all emails to build dictionary.
- Read full email content.
- Keep cleaning rules: alphabet-only words and remove single-letter words
- Keep top 3000 words.
- Use Multinomial Naive Bayes.
- Use word-to-index mapping for faster feature extraction.
- Use safe filename extraction (os.path.basename).
- Train model and report accuracy.