1. Importing Required Libraries

This section imports the essential libraries used for data manipulation, text preprocessing, and model development. NumPy and pandas handle data management, while scikit-learn and NLTK support text processing and machine learning tasks.

In [65]:
```
import numpy as np
import pandas as pd

from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline

from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from string import punctuation
```

2. Loading the Dataset

The dataset containing email text and corresponding spam labels is loaded for further preprocessing and model training.

In [2]:
```
df = pd.read_csv("data/emails.csv")
df.head()
```

Out[2]:

	text	spam
0	Subject: naturally irresistible your corporate...	1
1	Subject: the stock trading gunslinger fanny i...	1
2	Subject: unbelievable new homes made easy im ...	1
3	Subject: 4 color printing special request add...	1
4	Subject: do not have money , get software cds ...	1

3. Splitting the Dataset

The dataset is divided into training and testing sets using an 80:20 ratio. Stratified sampling ensures that both sets maintain the same proportion of spam and non-spam (ham) labels.

In [3]:
```
X = df.text
y = df.spam

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y, random_state=42)
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[3]:
```
((4582,), (1146,), (4582,), (1146,))
```

4. Model Training

In this stage, a machine learning pipeline is constructed using TF-IDF for text vectorization and Random Forest as the classifier. RandomizedSearchCV is used to optimize hyperparameters through a randomized search strategy.

In [66]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import RandomizedSearchCV
from jcopml.tuning import random_search_params as rsp

from sklearn.feature_extraction.text import TfidfVectorizer
```

In [42]:
```
pipeline = Pipeline([
    ('prep', TfidfVectorizer(tokenizer=word_tokenize, stop_words="english")),
    ('algo', RandomForestClassifier(n_jobs=-1, random_state=42))
])

model = RandomizedSearchCV(pipeline, rsp.rf_params, cv=3, n_iter=20, n_jobs=-1, verbose=1000, random_state=42)
model.fit(X_train, y_train)

print(model.best_params_)
print(model.score(X_train, y_train), model.best_score_, model.score(X_test, y_test))
```

```
Fitting 3 folds for each of 20 candidates, totalling 60 fits
{'algo__max_depth': 22, 'algo__max_features': 0.1185260448662222, 'algo__min_samples_leaf': 2, 'algo__n_estimators': 187}
0.9967263203841118 0.9783932838916947 0.9729493891797557
```

4. Sanity Check (Model Prediction Test)

Sample email texts are passed into the trained model to verify prediction performance. The model returns both the predicted class (spam/ham) and the probability distribution.

In [59]:
```
spam_text = ["""Hi,
We're excited to bring you the latest news from Paperpal!
In this update, you'll find out how the upgraded Write tool simplifies one of academic writing's biggest challenges: connecting ideas across a mix of files and formats.
Reference formatting on MS Word has also become a whole lot easier and faster. Read on for all the details.
Before you dive in, a quick reminder that our Big Black Friday Sale is live right now, with up to 60% off end-to-end academic writing support. It’s a great time to upgrade and get more out of Paperpal!"""]
```

In [68]:
```
model.predict(spam_text), model.predict_proba(spam_text)
```

Out[68]:
```
(array([1], dtype=int64), array([[0.20701669, 0.79298331]]))
```

In [63]:
```
ham_text = ["""Congratulations on being accepted at our office. You will start work tomorrow, Monday."""]
```

In [64]:
```
model.predict(ham_text), model.predict_proba(ham_text)
```

```
Out[64]:  (array([0], dtype=int64), array([[0.72258981, 0.27741019]]))
```