Loading Dataset

```
In [1]: import pandas as pd

        # Download the dataset
        data = pd.read_csv("C:/Users/sheej/Downloads/nlp_dataset.csv")

        # Display the first few rows of the dataset
        data.head()
```

Out[1]:

	Comment	Emotion
0	i seriously hate one subject to death but now ...	fear
1	im so full of life i feel appalled	anger
2	i sit here to write i start to dig out my feel...	fear
3	ive been really angry with r and i feel like a...	joy
4	i feel suspicious if there is no one outside l...	fear

Preprocessing

Text Cleaning, Tokenization, and Removal of Stopwords:

Text preprocessing typically involves:

Text Cleaning: Removing unwanted characters, URLs, special symbols, etc.

Tokenization: Splitting text into words or tokens.

Removal of Stopwords: Removing common words that don't contribute much to meaning.

In [2]:

```
import pandas as pd
import re
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import nltk

nltk.download('stopwords')
nltk.download('punkt')

# Load the dataset
url = 'https://drive.google.com/uc?id=1HWczIICsMpaL8EJyu48ZvRFcXx3_pcnb'
data = pd.read_csv(url)

# Display the column names and first few rows to confirm
print("Columns:", data.columns)
print("First few rows:\n", data.head())

# Preprocessing function
stop_words = set(stopwords.words('english'))

def preprocess_text(text):
    # Lowercase the text
    text = text.lower()
    # Remove URLs, special characters, and numbers
    text = re.sub(r'http\S+|www\S+|https\S+|[^a-zA-Z\s]', '', text, flags=r
    # Tokenization
    tokens = word_tokenize(text)
    # Remove stopwords
    tokens = [word for word in tokens if word not in stop_words]
    # Join tokens back to string
    return ' '.join(tokens)

# Apply preprocessing to the COMMENT column
data['clean_comment'] = data['Comment'].apply(preprocess_text)

# Display the cleaned text
print(data[['Comment', 'clean_comment']].head())
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\sheej\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\sheej\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

```
Columns: Index(['Comment', 'Emotion'], dtype='object')
First few rows:
                                              Comment Emotion
0  i seriously hate one subject to death but now ...    fear
1                 im so full of life i feel appalled   anger
2  i sit here to write i start to dig out my feel...    fear
3  ive been really angry with r and i feel like a...     joy
4  i feel suspicious if there is no one outside l...    fear
                                             Comment  \
0  i seriously hate one subject to death but now ...
1                 im so full of life i feel appalled
2  i sit here to write i start to dig out my feel...
3  ive been really angry with r and i feel like a...
4  i feel suspicious if there is no one outside l...

                                       clean_comment
0  seriously hate one subject death feel reluctan...
1                         im full life feel appalled
2  sit write start dig feelings think afraid acce...
3  ive really angry r feel like idiot trusting fi...
4  feel suspicious one outside like rapture happe...
```

Feature Extraction

We can use TfidfVectorizer to convert the text data into numerical features.

In [3]:

```python
from sklearn.feature_extraction.text import TfidfVectorizer

# Initialize TfidfVectorizer
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(data['clean_comment'])

# Display the shape of the feature matrix
print("Feature matrix shape:", X.shape)
```

```
Feature matrix shape: (5937, 8813)
```

Model Development

Train-Test Split and Model Training

We can split the data into training and testing sets, and then train the Naive Bayes and Support Vector Machine models.

In [4]:

```
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn.metrics import classification_report

# Extract features and target variable
X = vectorizer.fit_transform(data['clean_comment'])
y = data['Emotion']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, ra

# Naive Bayes Model
nb_model = MultinomialNB()
nb_model.fit(X_train, y_train)
nb_predictions = nb_model.predict(X_test)
print("Naive Bayes Classification Report:\n", classification_report(y_test,

# Support Vector Machine Model
svm_model = SVC()
svm_model.fit(X_train, y_train)
svm_predictions = svm_model.predict(X_test)
print("SVM Classification Report:\n", classification_report(y_test, svm_pre
```

```
Naive Bayes Classification Report:
               precision    recall  f1-score   support

       anger       0.87      0.94      0.91       600
        fear       0.93      0.88      0.91       614
         joy       0.91      0.89      0.90       568

    accuracy                           0.90      1782
   macro avg       0.91      0.90      0.90      1782
weighted avg       0.91      0.90      0.90      1782

SVM Classification Report:
               precision    recall  f1-score   support

       anger       0.93      0.93      0.93       600
        fear       0.97      0.87      0.92       614
         joy       0.87      0.98      0.92       568

    accuracy                           0.92      1782
   macro avg       0.93      0.92      0.92      1782
weighted avg       0.93      0.92      0.92      1782
```

Comparing model performance

After obtaining the classification reports, we can compare their metrics.

SVM model consistently shows higher accuracy, precision, recall, and F1-scores across different classes compared to the Naive Bayes model.

So SVM model is considered better.

In [ ]: