In [9]:
```
import pandas as pd
import re
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
```

In [10]:
```
data = pd.read_csv("C:/Users/999ra/Downloads/nlp_dataset.csv")
data
```

Out[10]:

	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
...	...	...
5932	i begun to feel distressed for you	fear
5933	i left feeling annoyed and angry thinking that...	anger
5934	i were to ever get married i d have everything...	joy
5935	i feel reluctant in applying there because i w...	fear
5936	i just wanted to apologize to you because i fe...	anger

5937 rows × 2 columns

In [11]:
```
x=data["Comment"]
y=data["Emotion"]
```

In [12]:
```
x
```

Out[12]:
```
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...
                                ...
5932                   i begun to feel distressed for you
5933    i left feeling annoyed and angry thinking that...
5934    i were to ever get married i d have everything...
5935    i feel reluctant in applying there because i w...
5936    i just wanted to apologize to you because i fe...
Name: Comment, Length: 5937, dtype: object
```

In [29]:
```
y
```

Out[29]:
```
0        fear
1       anger
2        fear
3         joy
4        fear
        ...
5932     fear
5933    anger
5934      joy
5935     fear
5936    anger
Name: Emotion, Length: 5937, dtype: object
```

In [38]:
```
def preprocess_text(text):
    text=text.lower() #convert to lowercase
    text=re.sub(r'[^a-zA-Z\s]','',text) #remove special characters
    tokens = word_tokenize(text) #tokenization
    stop_words = set(stopwords.words('english')) #remove stopwords
    tokens = [word for word in tokens if word not in stop_words]
    return ' '.join(tokens)
x1= data["Comment"].apply(preprocess_text)
```

In [39]:
```
from sklearn.feature_extraction.text import CountVectorizer
# Feature extraction
cv = CountVectorizer()
x1 = cv.fit_transform(x)
```

In [40]:
```
x1
```

Out[40]:
```
<5937x8954 sparse matrix of type '<class 'numpy.int64'>'
        with 93020 stored elements in Compressed Sparse Row format>
```

In [41]:
```
x1.toarray()
```

Out[41]:
```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

In [42]:
```
cv.get_feature_names_out()
```

Out[42]:
```
array(['aa', 'aac', 'aaron', ..., 'zonisamide', 'zq', 'zumba'],
      dtype=object)
```

In [43]:
```
import pandas as pd
pd.DataFrame(x1.toarray(),columns=cv.get_feature_names_out())
```

Out[43]:

	aa	aac	aaron	ab	abandon	abandoned	abandonment	abbigail	abc	abdomen	...	zendikar	zero	zest	zhu	zipline	zombie
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
5932	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
5933	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
5934	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
5935	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
5936	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	

5937 rows × 8954 columns

In [44]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x1, y, test_size=0.2)
```

In [45]:
```
x_train
```

Out[45]:
```
<4749x8954 sparse matrix of type '<class 'numpy.int64'>'
	with 74504 stored elements in Compressed Sparse Row format>
```

In [46]:
```
x_test
```

Out[46]:
```
<1188x8954 sparse matrix of type '<class 'numpy.int64'>'
	with 18516 stored elements in Compressed Sparse Row format>
```

In [47]:
```
x_train.shape
```

Out[47]:
```
(4749, 8954)
```

In [28]:
```
x_test.shape
```

Out[28]:
```
(1188, 8954)
```

In [48]:
```
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
```

In [49]:
```
# Train Naive Bayes
nb_model = MultinomialNB()
nb_model.fit(X_train, y_train)

# Train SVM
svm_model = SVC()
svm_model.fit(X_train, y_train)
```

Out[49]:
```
▾ SVC
SVC()
```

In [56]:
```
y_predict=nb_model.predict(x_test)
y_predict
```

Out[56]:
```
array(['anger', 'joy', 'joy', ..., 'anger', 'joy', 'joy'], dtype='<U5')
```

In [57]:
```
y1_predict=svm_model.predict(x_test)
y1_predict
```

Out[57]:
```
array(['joy', 'joy', 'joy', ..., 'anger', 'joy', 'anger'], dtype=object)
```

In [58]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_predict)
```

Out[58]:
```
0.335016835016835
```

In [59]:
```
accuracy_score(y_test,y1_predict)
```

Out[59]:
```
0.3122895622895623
```

In [66]:
```
z=["i begun to feel distressed for you"]
prediction=cv.transform(z)
```

In [67]:
```
nb_model.predict(prediction)
```

Out[67]:
```
array(['fear'], dtype='<U5')
```

In [68]:
```
svm_model.predict(prediction)
```

Out[68]:
```
array(['fear'], dtype=object)
```

In [ ]:
```
#EXPLANATION:
# Lowercasing: Converts all text to lowercase to ensure uniformity.
# Special Character Removal: Eliminates noise, allowing the model to focus on relevant words.
# Tokenization: Splits the text into individual words, essential for analysis.
# Stopword Removal: Filters out common words that may not carry significant meaning (e.g., "the," "is"). This
# countvectorizer: This method transforms the text data into a numerical format where each word is represente
# Accuracy: Measures the proportion of correctly classified instances.
# F1 Score: Provides a balance between precision and recall, making it more suitable for imbalanced datasets
#SUMMERY:
# loaded and preprocessed text data, extracted features using countvectorizer, developed and trained Naive Ba
# and compared their performance using accuracy and F1 score metrics. Each step is critical for building a ro
# emotion classification model.
```