Formative Assessment: NLP - Emotion Classification in Text#

In [ ]:

```
#1. Loading and Preprocessing
```

In [1]:

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

```

In [2]:

```

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


```

In [3]:

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


```

```
                                             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
```

In [6]:

```
# Preprocessing Function
def preprocess_text(text):
    # Remove special characters and digits
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    # Convert to lowercase
    text = text.lower()
    # Tokenization
    tokens = word_tokenize(text)
    # Remove stopwords
    stop_words = set(stopwords.words('english'))
    tokens = [word for word in tokens if word not in stop_words]
    return ' '.join(tokens)


```

In [9]:

```
print(data.columns)

```

```
Index(['Comment', 'Emotion'], dtype='object')
```

In [12]:

```python
data['cleaned_text'] = data['Comment'].apply(preprocess_text)
```

In [ ]:

```python
#2. Feature Extraction
```

In [13]:

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

In [14]:

```python
# Initialize TfidfVectorizer
vectorizer = TfidfVectorizer()
```

In [18]:

```python
# Fit and transform the cleaned text
X = vectorizer.fit_transform(data['cleaned_text'])
y = data['Emotion']
```

In [ ]:

```python
#3. Model Development
```

In [19]:

```python
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
```

In [20]:

```python
# Split the dataset into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, r
```

In [21]:

```python
# Naive Bayes model
nb_model = MultinomialNB()
nb_model.fit(X_train, y_train)
```

Out[21]:

```
▾ MultinomialNB

MultinomialNB()
```

In [22]:

```python
# SVM model
svm_model = SVC(kernel='linear')  # Linear kernel is generally effective f
svm_model.fit(X_train, y_train)

```

Out[22]:

```
▾ SVC
SVC(kernel='linear')
```

In [ ]:

```python
#4. Model Comparison
```

In [23]:

```python
from sklearn.metrics import accuracy_score, f1_score


```

In [24]:

```python
# Predictions
nb_predictions = nb_model.predict(X_test)
svm_predictions = svm_model.predict(X_test)


```

In [25]:

```python
# Calculate metrics
nb_accuracy = accuracy_score(y_test, nb_predictions)
nb_f1 = f1_score(y_test, nb_predictions, average='weighted')

svm_accuracy = accuracy_score(y_test, svm_predictions)
svm_f1 = f1_score(y_test, svm_predictions, average='weighted')


```

In [26]:

```python
# Display results
print(f"Naive Bayes - Accuracy: {nb_accuracy:.2f}, F1-Score: {nb_f1:.2f}")
print(f"SVM - Accuracy: {svm_accuracy:.2f}, F1-Score: {svm_f1:.2f}")

```

```
Naive Bayes - Accuracy: 0.91, F1-Score: 0.91
SVM - Accuracy: 0.95, F1-Score: 0.95
```

In [ ]:

```python

```