TEXT PROCESSING WITH NAIVE BAYES & SVM MODELS

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
```

```
data=pd.read_csv('/content/nlp_dataset.csv')
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

```
data.describe()
```

	Comment	Emotion
count	5937	5937
unique	5934	3
top	i feel like a tortured artist when i talk to her	anger
freq	2	2000

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5937 entries, 0 to 5936
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   Comment  5937 non-null   object
 1   Emotion  5937 non-null   object
dtypes: object(2)
memory usage: 92.9+ KB
```

```
data.isnull().sum()
```

	0
Comment	0
Emotion	0

dtype: int64

```
data.columns
```

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

```
data['Emotion'].value_counts()
```

Emotion	count
anger	2000
joy	2000
fear	1937

dtype: int64

```
data.shape
```

```
(5937, 2)
```

```
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
True
```

```
!pip install nltk
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.8.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk) (2024.9.11)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk) (4.66.5)
```

```
nltk.download('punkt_tab')
from nltk.tokenize import word_tokenize
```

```
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
```

```
nltk.download('wordnet')
nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
True
```

TOKENIZATION - This process splits the text into indivual words.

LOWERCASE - This process converts all the text which contains caps to lowercase

```
from nltk.tokenize import word_tokenize

clean_t = []

for text in data['Comment']:

    tokens = word_tokenize(text)
    filtered_tokens = [lemmatizer.lemmatize(i.lower()) for i in tokens if i.lower() not in stop_words and i not in string.punctuation]
    clean_t.append(' '.join(filtered_tokens))

data['clean_t'] = clean_t
```

STOP WORDS REMOVAL - It helps to remove the common words in the data like (is , and ..) which word doesnt provide any significant meaning.

```
from nltk.corpus import stopwords
nltk.download('stopwords')

stop_words = set(stopwords.words('english'))
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

LEMMATIZATION - It helps to reduces words to their dictinoary form

```
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()
```

TfidfVectorizer- When we convert text data into numbers, we create a matrix that shows how important each word is based on how often it appears in different messages. Each row in this matrix represents a single message, and each column represents a unique word. This way, we can analyze the text using numbers.

```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(data['clean_t'])
y = data['Emotion']


from sklearn.model_selection import train_test_split


X_train, X_test, y_train, y_test = train_test_split(X, y)
```

NAIVE BAYES -This is a type of algorithm that uses probability to classify text. It assumes that each word in a message is independent of the others, which makes it effective for tasks like email filtering and sentiment analysis.

```
from sklearn.naive_bayes import MultinomialNB


nb= MultinomialNB()
nb.fit(X_train, y_train)
```

```
MultinomialNB
MultinomialNB()
```

```
from sklearn.metrics import accuracy_score, f1_score


y_pred_nb = nb.predict(X_test)
accuracy = accuracy_score(y_test, y_pred_nb)
f1 = f1_score(y_test, y_pred_nb, average='weighted')
print("NAVIE BAYES MODEL - \n","Accuracy:", accuracy)
print("F1 Score:", f1)
```

```
NAVIE BAYES MODEL -
 Accuracy: 0.9151515151515152
F1 Score: 0.9151740608109338
```

SUPPORT VECTOR MACHINE (SVM) - This is a strong classification method that can handle both straight-line (linear) and more complex (non-linear) relationships in the data. It works well for a variety of classification tasks.

```
from sklearn.svm import SVC


svm = SVC()
svm.fit(X_train, y_train)
```

```
SVC
SVC()
```

```
y_pred_svm = svm.predict(X_test)
accuracy = accuracy_score(y_test, y_pred_svm)
f1 = f1_score(y_test, y_pred_svm, average='weighted')
print("SVM MODEL - \n","Accuracy:",accuracy)
print("F1 Score:", f1)
```

```
SVM MODEL -
 Accuracy: 0.9306397306397306
F1 Score: 0.9305987487604515
```

Accuracy: This metric tells us how many messages were classified correctly . It gives a idea of how well the model is performing.

F1-Score: precision-how many of the predicted were actually positive .

recall -how many of the actual positive cases were correctly predicted.

```
Start coding or generate with AI.
```