```
import pandas as pd
```

```
df=pd.read_csv("/content/nlp_dataset (1).csv")
df
```

	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

Next steps: Generate code with df View recommended plots New interactive sheet

```
x=df['Comment']
y=df['Emotion']
```

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

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
True
```

DATA PREPROCESSING

1.Tokenizaion

```
x
```

```
                                                 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...
  ...                                                        ...
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...

5937 rows × 1 columns

dtype: object
```

```
from nltk.tokenize import word_tokenize
x_tokenized=x.apply(word_tokenize)
x_tokenized
```

```
                                          Comment
   0      [i, seriously, hate, one, subject, to, death, ...
   1                 [im, so, full, of, life, i, feel, appalled]
   2               [i, sit, here, to, write, i, start, to, dig, o...
   3             [ive, been, really, angry, with, r, and, i, fe...
   4              [i, feel, suspicious, if, there, is, no, one, ...
  ...                                                        ...
5932                 [i, begun, to, feel, distressed, for, you]
5933      [i, left, feeling, annoyed, and, angry, thinki...
5934       [i, were, to, ever, get, married, i, d, have, ...
5935       [i, feel, reluctant, in, applying, there, beca...
5936    [i, just, wanted, to, apologize, to, you, beca...
5937 rows × 1 columns

dtype: object
```

2.Lower case Tokens

```
x_lowercase=x_tokenized.apply(lambda tokens:[token.lower() for token in tokens])
x_lowercase
```

	Comment
0	[i, seriously, hate, one, subject, to, death, ...
1	[im, so, full, of, life, i, feel, appalled]
2	[i, sit, here, to, write, i, start, to, dig, o...
3	[ive, been, really, angry, with, r, and, i, fe...
4	[i, feel, suspicious, if, there, is, no, one, ...
...	...
5932	[i, begun, to, feel, distressed, for, you]
5933	[i, left, feeling, annoyed, and, angry, thinki...
5934	[i, were, to, ever, get, married, i, d, have, ...
5935	[i, feel, reluctant, in, applying, there, beca...
5936	[i, just, wanted, to, apologize, to, you, beca...

5937 rows × 1 columns

dtype: object

3.Stopwords Removal

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

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

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

```
 'while',
 'who',
 'whom',
 'why',
 'will',
 'with',
 'won',
 "won't",
 'wouldn',
 "wouldn't",
 'y',
 'you',
 "you'd",
 "you'll",
 "you're",
 "you've",
 'your',
 'yours',
 'yourself',
 'yourselves'}
```

```
x_stop_words=x_lowercase.apply(lambda i:[word for word in i if word not in stop_words])
x_stop_words
```

	Comment
0	[seriously, hate, one, subject, death, feel, r...
1	[im, full, life, feel, appalled]
2	[sit, write, start, dig, feelings, think, afra...
3	[ive, really, angry, r, feel, like, idiot, tru...
4	[feel, suspicious, one, outside, like, rapture...
...	...
5932	[begun, feel, distressed]
5933	[left, feeling, annoyed, angry, thinking, cent...
5934	[ever, get, married, everything, ready, offer,...
5935	[feel, reluctant, applying, want, able, find, ...
5936	[wanted, apologize, feel, like, heartless, bitch]

5937 rows × 1 columns

dtype: object

LEMMATIZATION

```
from nltk.stem import WordNetLemmatizer
nltk.download('wordnet')
nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
True
```

```
lemmatizer=WordNetLemmatizer()
lemmatized_tokens=x_stop_words.apply(lambda i:[lemmatizer.lemmatize(word) for word in i])
lemmatized_tokens
```

	Comment
0	[seriously, hate, one, subject, death, feel, r...
1	[im, full, life, feel, appalled]
2	[sit, write, start, dig, feeling, think, afrai...
3	[ive, really, angry, r, feel, like, idiot, tru...
4	[feel, suspicious, one, outside, like, rapture...
...	...
5932	[begun, feel, distressed]
5933	[left, feeling, annoyed, angry, thinking, cent...
5934	[ever, get, married, everything, ready, offer,...
5935	[feel, reluctant, applying, want, able, find, ...
5936	[wanted, apologize, feel, like, heartless, bitch]

5937 rows × 1 columns

dtype: object

Avoid punctuation

```
import string
string.punctuation
```

```
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

```
tokens_no_punct=lemmatized_tokens.apply(lambda i:[word for word in i if word not in string.punctuation])
tokens_no_punct
```

```
                                               Comment
   0      [seriously, hate, one, subject, death, feel, r...
   1                       [im, full, life, feel, appalled]
   2            [sit, write, start, dig, feeling, think, afrai...
   3             [ive, really, angry, r, feel, like, idiot, tru...
   4       [feel, suspicious, one, outside, like, rapture...
  ...                                                   ...
 5932                             [begun, feel, distressed]
 5933    [left, feeling, annoyed, angry, thinking, cent...
 5934    [ever, get, married, everything, ready, offer,...
 5935      [feel, reluctant, applying, want, able, find, ...
 5936   [wanted, apologize, feel, like, heartless, bitch]

5937 rows × 1 columns

dtype: object
```

Feature Extraction

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer=CountVectorizer()
x_vectorized=vectorizer.fit_transform(x)
```

```
from sklearn.model_selection import train_test_split


x_train,x_test,y_train,y_test=train_test_split(x_vectorized,y,test_size=0.2,random_state=42)
```

MODEL Development

Naive Bayes Model

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix


nb_model=MultinomialNB()
nb_model.fit(x_train,y_train)

y_pred_nb=nb_model.predict(x_test)


print("Naive Bayes Accuracy:",accuracy_score(y_test,y_pred_nb))
print("Classification Report for Naive Bayes:\n",classification_report(y_test,y_pred_nb))
```

```
Naive Bayes Accuracy: 0.8939393939393939
Classification Report for Naive Bayes:
               precision    recall  f1-score   support

       anger       0.88      0.92      0.90       392
        fear       0.88      0.92      0.90       416
         joy       0.92      0.84      0.88       380

    accuracy                           0.89      1188
   macro avg       0.90      0.89      0.89      1188
weighted avg       0.89      0.89      0.89      1188
```

Support Vector Machine Model

```
from sklearn.svm import SVC
svm_model=SVC(kernel="linear")
svm_model.fit(x_train,y_train)
y_pred_svm=svm_model.predict(x_test)
```

```
print("SVM Accuracy:",accuracy_score(y_test,y_pred_svm))
print("Classification Report for SVM:\n",classification_report(y_test,y_pred_svm))
```

```
SVM Accuracy: 0.9486531986531986
Classification Report for SVM:
               precision    recall  f1-score   support

       anger       0.92      0.96      0.94       392
        fear       0.97      0.92      0.95       416
         joy       0.96      0.96      0.96       380

    accuracy                           0.95      1188
   macro avg       0.95      0.95      0.95      1188
weighted avg       0.95      0.95      0.95      1188
```

```
# confusion matrix for Naive Bayes

print("confusion matrix for Naive Bayes:\n",confusion_matrix(y_test,y_pred_nb))

#Confusion matrix for SVM
print("Confusion matrix for SVM:\n",confusion_matrix(y_test,y_pred_svm))

#Accuracy and F1 score
print("Naive Bayes Accuracy:",accuracy_score(y_test,y_pred_nb))
print("SVM Accuracy:",accuracy_score(y_test,y_pred_svm))
```

```
confusion matrix for Naive Bayes:
 [[359  21  12]
 [ 18 382  16]
 [ 30  29 321]]
Confusion matrix for SVM:
 [[377   7   8]
 [ 24 384   8]
 [  9   5 366]]
Naive Bayes Accuracy: 0.8939393939393939
SVM Accuracy: 0.9486531986531986
```

Double-click (or enter) to edit

Double-click (or enter) to edit