```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

importing necessary libraries

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

step 1 : load the dataset into a dataframe

```
df = pd.read_csv('/content/Emotion_final.csv')
df.head()
```

	Text	Emotion
0	i didnt feel humiliated	sadness
1	i can go from feeling so hopeless to so damned...	sadness
2	im grabbing a minute to post i feel greedy wrong	anger
3	I am in love with you	love
4	i am ever feeling nostalgic about the fireplac...	love

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

step 2 : Perform the data cleaning

```
# summary of dataframe
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21493 entries, 0 to 21492
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   Text     21493 non-null  object
 1   Emotion  21493 non-null  object
dtypes: object(2)
memory usage: 336.0+ KB
```

```
df.describe()
```

	Text	Emotion
count	21493	21493
unique	21439	7
top	i often find myself feeling assaulted by a mul...	happy
freq	2	7033

a . handle missing values

```
# checking for missing values
df.isnull().sum()
```

```
Text       0
Emotion    0
dtype: int64
```

```
df.isnull().sum()/len(df)
```

```
Text       0.0
Emotion    0.0
dtype: float64
```

b.remove duplicates

```
# drop rows with missing values
df.dropna(subset=['Text', 'Emotion'], inplace = True)
```

```
# check duplicate rows
df.duplicated().sum()
```

```
3
```

```
# remove duplicate rows
df.drop_duplicates(inplace=True)
```

```
# Reset index after dropping rows
df.reset_index(drop=True, inplace=True)
```

step 3 : Label encoding the emotion column

```
from sklearn.preprocessing import LabelEncoder
lbec = LabelEncoder()            # initialize label encoder

lbec.fit(df['Emotion'])
Emotion_encoded = lbec.transform(df['Emotion'])
Emotion_encoded
```

```
array([5, 5, 0, ..., 1, 1, 1])
```

```
Emotion_encoded[:5]
```

```
array([5, 5, 0, 4, 4])
```

```
label_mapping = dict(zip(lbec.classes_, lbec.transform(lbec.classes_)))
print("Label Mapping:")
print(label_mapping)             # mapping original labels to encoded values
```

```
Label Mapping:
{'anger': 0, 'confusion': 1, 'fear': 2, 'happy': 3, 'love': 4, 'sadness': 5, 'surprise': 6}
```

step 4 : train a random forest model with the dataset

```
# define x and y variables
X = df['Text']
Y = df['Emotion']
```

```
#split a data set into train and test
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test = train_test_split(X, Y, train_size = 0.2, random_state = 42)
```

```
from sklearn.ensemble import RandomForestClassifier
```

```
from sklearn.feature_extraction.text import TfidfVectorizer

# Create a TfidfVectorizer to convert text to numerical features
vectorizer = TfidfVectorizer()

# Fit the vectorizer on the training data and transform both training and testing data
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)

# Now fit the model with the vectorized data
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train_vec, Y_train)
```

```
▾        RandomForestClassifier
RandomForestClassifier(random_state=42)
```

Step 5: Find the accuracy of the model

```
# predictions
Y_pred = model.predict(vectorizer.transform(X_test)) # Transform x_test before prediction
```

```
from sklearn.metrics import accuracy_score
# evaluating the model
accuaracy = accuracy_score(Y_test, Y_pred)
print("accuracy:" , accuaracy)
```

```
accuracy: 0.7259771986970684
```

```
from sklearn.metrics import confusion_matrix, classification_report

# Calculate confusion matrix (use a different variable name)
conf_matrix_result = confusion_matrix(Y_test, Y_pred)
print("confusion_matrix:" , conf_matrix_result)
```

```
confusion_matrix: [[1579    0   75  440    4  274    3]
 [   0    0    2    6    0    3    0]
 [  54    0 1298  483   10  258   28]
 [  43    0   43 5060   58  421   14]
 [  19    0    7  622  570  111    2]
 [  89    0   81 1133   13 3675    8]
 [  14    0  108  186    2   97  299]]
```

```
#classification_report

c_report = classification_report(Y_test, Y_pred)
print("classification_report:" , c_report)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1471: UndefinedMetricWarning: Precision and F-score are
  _warn_prf(average, modifier, msg_start, len(result))
classification_report:               precision    recall  f1-score   support

       anger       0.88      0.66      0.76      2375
   confusion       0.00      0.00      0.00        11
        fear       0.80      0.61      0.69      2131
       happy       0.64      0.90      0.75      5639
        love       0.87      0.43      0.57      1331
     sadness       0.76      0.74      0.75      4999
    surprise       0.84      0.42      0.56       706

    accuracy                           0.73     17192
   macro avg       0.68      0.54      0.58     17192
weighted avg       0.75      0.73      0.72     17192

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1471: UndefinedMetricWarning: Precision and F-score are
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1471: UndefinedMetricWarning: Precision and F-score are
  _warn_prf(average, modifier, msg_start, len(result))
```