Project on sentiment analyzer

data set :
https://drive.google.com/file/d/14BM6X1YkPb3r391kXtONYzvWryFuGpOF/view?usp=drive_link

1: Load the dataset into a Dataframe

```
import pandas as pd
```

```
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 recommended interactive

2: Perform the data cleaning

```
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
```

```
print(df.isnull())
```

```
        Text  Emotion
0      False    False
1      False    False
2      False    False
3      False    False
4      False    False
...      ...      ...
21488  False    False
21489  False    False
21490  False    False
21491  False    False
21492  False    False

[21493 rows x 2 columns]
```

```
print(df.isnull().sum())
```

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

```
df.dropna(subset=['Text', 'Emotion'], inplace=True)
```

```
print("After handling missing values:")
print(df.isnull().sum())
```

```
After handling missing values:
Text       0
Emotion    0
dtype: int64
```

```
print("Duplicate rows:", df.duplicated().sum())

# Remove duplicates
df.drop_duplicates(inplace=True)

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

```
Duplicate rows: 3
```

3: Label encode the Emotion column

```
from sklearn.preprocessing import LabelEncoder
```

```
# Initialize LabelEncoder
label_encoder = LabelEncoder()
```

```
# Encode labels in column 'Emotion'
df['Emotion_encoded'] = label_encoder.fit_transform(df['Emotion'])
```

```
# Display the mapping of original labels to encoded values
label_mapping = dict(zip(label_encoder.classes_, label_encoder.tra
print("Label Mapping:")
print(label_mapping)
```

```
Label Mapping:
{'anger': 0, 'confusion': 1, 'fear': 2, 'happy': 3, 'love': 4,
```

4: Train a random forest model with the dataset

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
```

```
# Split the data into features and target
X = df['Text']
y = df['Emotion']
```

```
# Convert text data to numerical data using TF-IDF Vectorizer
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer(max_features=1000)
X = tfidf.fit_transform(X).toarray()
```

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_siz
```

```
# Initialize and train the Random Forest Classifier
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

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

5: Find the accuracy of the model

```
from sklearn.metrics import accuracy_score
```

```
# Predict the labels for the test set
y_pred = model.predict(X_test)
```

```
accuracy = accuracy_score(y_test, y_pred)
print(accuracy)
```

```
0.8089809213587715
```

```
# Print the accuracy in percentage
print("Model accuracy: {:.2f}%".format(accuracy * 100))
```

```
Model accuracy: 80.90%
```

```
print("Model Accuracy: {:.2f}".format(accuracy))
```

```
Model Accuracy: 0.81
```

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

Disk 80.38 GB availab