```
# === FULL EMOTION DATA ANALYSIS & CLASSIFICATION PIPELINE ===

# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
import string

from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import LabelEncoder
import joblib

# === Step 1: Load the dataset ===
df = pd.read_csv("/content/sentimentdataset.csv")
print("\nFirst 5 rows of the dataset:")
print(df.head())

# === Check column names to confirm ===
print("\nColumn names:", df.columns.tolist())
```

```
# === Step 2: Explore the dataset ===
print("\nDataset info:")
print(df.info())

print("\nDataset shape:", df.shape)

print("\nChecking for missing values:")
print(df.isnull().sum())



# === Step 3: Basic EDA ===
plt.figure(figsize=(10,6))
sns.countplot(x='Sentiment', data=df)   # <-- REPLACED 'Emotion' with 'Sentiment'
plt.title('Distribution of Sentiments')
plt.xticks(rotation=45)
plt.show()

print("\nClass distribution:")
print(df['Sentiment'].value_counts())   # <-- REPLACED 'Emotion' with 'Sentiment'
```

```
# === Step 5: Encode labels ===
le = LabelEncoder()
df['Encoded_Sentiment'] = le.fit_transform(df['Sentiment'])  # <-- REPLACED 'Emotion' with 'Sentiment'
print("\nEncoded sentiments:")
print(dict(zip(le.classes_, le.transform(le.classes_))))

# === Step 6: Build the model pipeline ===
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression

model_pipeline = Pipeline([
    ('tfidf', TfidfVectorizer(max_features=5000, ngram_range=(1,2))),
    ('clf', LogisticRegression(max_iter=1000, solver='lbfgs'))
])

print("\nTraining the model...")
```

```
#HEATMAP
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
import re

# Load CSV
df = pd.read_csv("/content/sentimentdataset.csv")

# Check columns
print("Available columns:", df.columns)

# Clean text
if 'Cleaned_Text' not in df.columns:
    def clean_text(text):
        text = str(text).lower()
        text = re.sub(r'\\[.*?\\]', '', text)
        text = re.sub(r'https?://\\S+|www\\.\\S+', '', text)
        text = re.sub(r'<.*?>+', '', text)
        text = re.sub(r'[^a-zA-Z\\s]', '', text)
        return text
    df['Cleaned_Text'] = df['Text'].apply(clean_text)

# Feature Engineering
df['Text_Length'] = df['Cleaned_Text'].apply(len)
df['Word_Count'] = df['Cleaned_Text'].apply(lambda x: len(x.split()))

# Encode label (replace 'Sentiment' with your actual label column name!!)
le = LabelEncoder()
df['Encoded_Emotion'] = le.fit_transform(df['Sentiment'])   # CHANGE THIS IF COLUMN IS DIFFERENT
```

```
# Compute correlation matrix
numeric_cols = ['Text_Length', 'Word_Count', 'Encoded_Emotion']
corr_matrix = df[numeric_cols].corr()
print("\nCorrelation Matrix:")
print(corr_matrix)

# Plot heatmap
plt.figure(figsize=(8,6))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Matrix Heatmap')
plt.show()
```

```
#BAR CHAT
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Example sentiment data
sentiment_scores = [-0.8, -0.3, 0.4, 0.7, 0.2, 1.0, -0.9, 0.3, -0.1, 0.6, 0.2, -0.6, 1.2, 0.5]

# Create a DataFrame
df = pd.DataFrame(sentiment_scores, columns=["Sentiment"])

# Create a box plot
plt.figure(figsize=(8, 6))
sns.boxplot(x=df["Sentiment"])

# Show plot
plt.show()
```

```
#Z SCORE
import numpy as np
# Load CSV first
df = pd.read_csv("/content/sentimentdataset.csv")

# Sample sentiment scores for the dataset (can be your sentiment data)

# Step 1: Calculate the mean and standard deviation
mean = np.mean(sentiment_scores)
std_dev = np.std(sentiment_scores)

# Step 2: Calculate the z-scores for each sentiment score
z_scores = [(x - mean) / std_dev for x in sentiment_scores]

# Output the z-scores
print("Sentiment Scores: ", sentiment_scores)
print("Z-scores: ", z_scores)
```