**# Upload the Dataset**

from google.colab import files

uploaded = files.upload()

**#Load the Dataset**

import pandas as pd

df = pd.read\_csv('sentimentdataset.csv')

df.columns = df.columns.str.strip()

df['Sentiment'] = df['Sentiment'].str.strip()

df.head()

**#Data Exploration**

df.info()

df.describe(include='all')

df['Sentiment'].value\_counts()ra0

**# Check for Missing Values and Duplicates**

print("Missing values:\n", df.isnull().sum())

print("Duplicates:", df.duplicated().sum())

# Drop duplicates if needed

df = df.drop\_duplicates()

**#Visualize a Few Features**

import seaborn as sns

import matplotlib.pyplot as plt

sns.countplot(x='Sentiment', data=df)

plt.title("Sentiment Distribution")plt.show()

**# Identify Target and Features**

X = df['Text'] # Input feature

y = df['Sentiment']

**#Convert Categorical Columns to Numerical**

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

y\_encoded = le.fit\_transform(y) # Positive, Negative, Neutral -> 2, 0, 1 (for example)

**#One-Hot Encoding**

# Optional: One-hot encode sentiment labels

y\_onehot = pd.get\_dummies(df['Sentiment'])

**# Feature Scaling (Text Vectorization using TF-IDF)**

from sklearn.feature\_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer(stop\_words='english', max\_df=0.7)

X\_tfidf = tfidf.fit\_transform(X)

**# Train-Test Split**

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_tfidf, y\_encoded, test\_size=0.2, random\_state=42)

**# Model Building**

from sklearn.naive\_bayes import MultinomialNB

model = MultinomialNB()

model.fit(X\_train, y\_train)

**# Evaluation**

from sklearn.metrics import classification\_report, accuracy\_score

# Get unique classes from y\_test and y\_pred

unique\_classes = sorted(list(set(y\_test) | set(y\_pred)))

# Print accuracy

print("Accuracy:", accuracy\_score(y\_test, y\_pred))

# Generate and print the classification report

print(classification\_report(y\_test, y\_pred, target\_names=[str(c) for c in unique\_classes]))

# Convert target\_names to strings to avoid warning

**#Make Predictions from New Input**

def predict\_sentiment(text):

vector = tfidf.transform([text])

pred = model.predict(vector)[0]

return le.inverse\_transform([pred])[0]

predict\_sentiment("I love this new update!")

np.int64(158)

**# Predict the Final Grade**

final\_accuracy = accuracy\_score(y\_test, y\_pred)

print(f"Model Grade: {final\_accuracy\*100:.2f}%")

**#Deployment — Building an Interactive App**

pip install gradio pandas scikit-learn

**# Create a Prediction Function**

def sentiment\_app(text):

vector = tfidf.transform([text])

prediction = model.predict(vector)[0]

**# Create the Gradio Interface**

!pip install gradio

import gradio as gr

interface = gr.Interface(

fn=sentiment\_app,

inputs="text",

outputs="text",

title="🗣 Social Media Sentiment Analyzer",

description="Enter a comment or post and get the predicted sentiment."

)

interface.launch(share=True)