```
# 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)
```