1. Upload the Dataset

```
from google.colab import files
uploaded = files.upload()
```

Choose Files sentimentdataset.csv

- **sentimentdataset.csv**(text/csv) - 170776 bytes, last modified: 5/15/2025 - 100% done

Saving sentimentdataset.csv to sentimentdataset.csv

2. 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()
```

| | Unnamed: 0.1 | Unnamed: 0 | Text | Sentiment | Timestamp | User | Platform | Hashtags | Retweets | Likes | Country | Year | Month | Day | H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 0 | Enjoying a beautiful day at the park! ... | Positive | 2023-01-15 12:30:00 | User123 | Twitter | #Nature #Park | 15.0 | 30.0 | USA | 2023 | 1 | 15 | |
| **1** | 1 | 1 | Traffic was terrible this morning. ... | Negative | 2023-01-15 08:45:00 | CommuterX | Twitter | #Traffic #Morning | 5.0 | 10.0 | Canada | 2023 | 1 | 15 | |

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

3. Data Exploration

```
df.info()
df.describe(include='all')
df['Sentiment'].value_counts()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 732 entries, 0 to 731
Data columns (total 15 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Unnamed: 0.1  732 non-null    int64
 1   Unnamed: 0    732 non-null    int64
 2   Text          732 non-null    object
 3   Sentiment     732 non-null    object
 4   Timestamp     732 non-null    object
 5   User          732 non-null    object
 6   Platform      732 non-null    object
 7   Hashtags      732 non-null    object
 8   Retweets      732 non-null    float64
 9   Likes         732 non-null    float64
 10  Country       732 non-null    object
 11  Year          732 non-null    int64
 12  Month         732 non-null    int64
 13  Day           732 non-null    int64
 14  Hour          732 non-null    int64
dtypes: float64(2), int64(6), object(7)
memory usage: 85.9+ KB
```

| Sentiment | count |
|---|---|
| **Positive** | 45 |
| **Joy** | 44 |
| **Excitement** | 37 |
| **Contentment** | 19 |
| **Neutral** | 18 |
| **...** | ... |
| **Celestial Wonder** | 1 |
| **Nature's Beauty** | 1 |
| **Thrilling Journey** | 1 |
| **Whispers of the Past** | 1 |
| **Relief** | 1 |

191 rows × 1 columns

**dtype:** int64

4. 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()
```

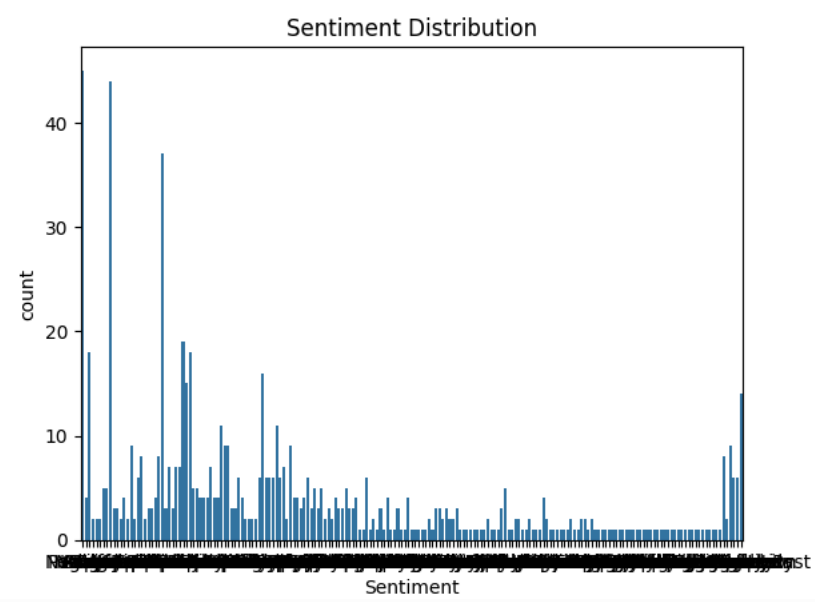
```
Missing values:
 Unnamed: 0.1    0
Unnamed: 0       0
Text             0
Sentiment        0
Timestamp        0
User             0
Platform         0
Hashtags         0
Retweets         0
Likes            0
Country          0
Year             0
Month            0
Day              0
Hour             0
dtype: int64
Duplicates: 0
```

5. 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()
```




6. Identify Target and Features

```
X = df['Text']  # Input feature
y = df['Sentiment']  # Target class
```

7. 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)
```

8. One Encoding / 8. One-Hot Encoding

```
# Optional: One-hot encode sentiment labels
y_onehot = pd.get_dummies(df['Sentiment'])
```

9. 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)
```

10. 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)
```

11. Model Building

```
from sklearn.naive_bayes import MultinomialNB

model = MultinomialNB()
model.fit(X_train, y_train)
```

```
MultinomialNB
MultinomialNB()
```

12. 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
```

```
              76       0.33      0.57      0.42         7
              79       0.00      0.00      0.00         1
              80       0.00      0.00      0.00         1
              85       0.00      0.00      0.00         1
              86       0.00      0.00      0.00         3
              87       0.00      0.00      0.00         2
              89       0.00      0.00      0.00         1
              91       0.00      0.00      0.00         1
              93       0.00      0.00      0.00         6
              95       0.00      0.00      0.00         2
              97       0.00      0.00      0.00         2
             101       1.00      1.00      1.00         1
             107       0.00      0.00      0.00         1
             108       0.00      0.00      0.00         1
             109       0.00      0.00      0.00         1
             112       0.00      0.00      0.00         1
             114       0.00      0.00      0.00         1
             116       0.20      0.89      0.32         9
             118       0.00      0.00      0.00         1
             119       0.00      0.00      0.00         1
             121       0.00      0.00      0.00         2
             123       0.00      0.00      0.00         1
             126       0.00      0.00      0.00         2
             130       0.00      0.00      0.00         1
             135       0.00      0.00      0.00         1
             136       0.00      0.00      0.00         2
             137       0.00      0.00      0.00         1
             142       0.00      0.00      0.00         1
             144       0.00      0.00      0.00         2
             146       0.06      0.67      0.12         9
             150       0.00      0.00      0.00         1
             152       0.00      0.00      0.00         1
             153       0.00      0.00      0.00         1
             158       0.00      0.00      0.00         1
             159       0.00      0.00      0.00         1
             164       0.00      0.00      0.00         2
             165       0.00      0.00      0.00         1
             166       0.00      0.00      0.00         4
             169       0.00      0.00      0.00         1
             170       0.00      0.00      0.00         1
             171       0.00      0.00      0.00         1
             174       0.00      0.00      0.00         1
             178       0.00      0.00      0.00         1
             183       0.00      0.00      0.00         1
             185       0.00      0.00      0.00         1
             190       0.00      0.00      0.00         1

        accuracy                           0.13       147
       macro avg       0.02      0.04      0.02       147
    weighted avg       0.04      0.13      0.05       147

/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

13. 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)
```

15. Predict the Final Grade

```
final_accuracy = accuracy_score(y_test, y_pred)
print(f"Model Grade: {final_accuracy*100:.2f}%")
```

```
Model Grade: 12.93%
```

16. Deployment — Building an Interactive App

```
pip install gradio pandas scikit-learn
```

```
Collecting tomlkit<0.14.0,>=0.12.0 (from gradio)
  Downloading tomlkit-0.13.2-py3-none-any.whl.metadata (2.7 kB)
Requirement already satisfied: typer<1.0,>=0.12 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.15.3)
Requirement already satisfied: typing-extensions~=4.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (4.13.2)
Collecting uvicorn>=0.14.0 (from gradio)
  Downloading uvicorn-0.34.2-py3-none-any.whl.metadata (6.5 kB)
Requirement already satisfied: fsspec in /usr/local/lib/python3.11/dist-packages (from gradio-client==1.10.1->gradio) (2025.3.2)
Requirement already satisfied: websockets<16.0,>=10.0 in /usr/local/lib/python3.11/dist-packages (from gradio-client==1.10.1->gra
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.2)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.15.3)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.5.0)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (3.6.0)
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.11/dist-packages (from anyio<5.0,>=3.0->gradio) (3.10)
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.11/dist-packages (from anyio<5.0,>=3.0->gradio) (1.3.1)
Requirement already satisfied: certifi in /usr/local/lib/python3.11/dist-packages (from httpx>=0.24.1->gradio) (2025.4.26)
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.11/dist-packages (from httpx>=0.24.1->gradio) (1.0.9)
Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.11/dist-packages (from httpcore==1.*->httpx>=0.24.1->gradio) (
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.28.1->gradio) (3.18.0
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.28.1->gradio) (2.32.3
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.28.1->gradio) (4.
Requirement already satisfied: hf-xet<2.0.0,>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.28.1->gra
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.11/dist-packages (from pydantic<2.12,>=2.0->gradi
Requirement already satisfied: pydantic-core==2.33.2 in /usr/local/lib/python3.11/dist-packages (from pydantic<2.12,>=2.0->gradio
Requirement already satisfied: typing-inspection>=0.4.0 in /usr/local/lib/python3.11/dist-packages (from pydantic<2.12,>=2.0->gra
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas) (1.17.0)
Requirement already satisfied: click>=8.0.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0,>=0.12->gradio) (8.1.8)
Requirement already satisfied: shellingham>=1.3.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0,>=0.12->gradio) (1.5
Requirement already satisfied: rich>=10.11.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0,>=0.12->gradio) (13.9.4)
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.11/dist-packages (from rich>=10.11.0->typer<1.0,>=
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist-packages (from rich>=10.11.0->typer<1.0,
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests->huggingface-hu
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests->huggingface-hub>=0.2
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-packages (from markdown-it-py>=2.2.0->rich>=10.11.0->
Downloading gradio-5.29.1-py3-none-any.whl (54.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 54.1/54.1 MB 14.4 MB/s eta 0:00:00
Downloading gradio_client-1.10.1-py3-none-any.whl (323 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 323.1/323.1 kB 18.7 MB/s eta 0:00:00
Downloading aiofiles-24.1.0-py3-none-any.whl (15 kB)
Downloading fastapi-0.115.12-py3-none-any.whl (95 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 95.2/95.2 kB 6.7 MB/s eta 0:00:00
Downloading groovy-0.1.2-py3-none-any.whl (14 kB)
Downloading python_multipart-0.0.20-py3-none-any.whl (24 kB)
Downloading ruff-0.11.9-py3-none-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (11.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11.5/11.5 MB 91.5 MB/s eta 0:00:00
Downloading safehttpx-0.1.6-py3-none-any.whl (8.7 kB)
Downloading semantic_version-2.10.0-py2.py3-none-any.whl (15 kB)
Downloading starlette-0.46.2-py3-none-any.whl (72 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 72.0/72.0 kB 4.2 MB/s eta 0:00:00
Downloading tomlkit-0.13.2-py3-none-any.whl (37 kB)
Downloading uvicorn-0.34.2-py3-none-any.whl (62 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 62.5/62.5 kB 3.9 MB/s eta 0:00:00
Downloading ffmpy-0.5.0-py3-none-any.whl (6.0 kB)
Downloading pydub-0.25.1-py2.py3-none-any.whl (32 kB)
Installing collected packages: pydub, uvicorn, tomlkit, semantic-version, ruff, python-multipart, groovy, ffmpy, aiofiles, starle
Successfully installed aiofiles-24.1.0 fastapi-0.115.12 ffmpy-0.5.0 gradio-5.29.1 gradio-client-1.10.1 groovy-0.1.2 pydub-0.25.1
```

17. Create a Prediction Function

```
def sentiment_app(text):
    vector = tfidf.transform([text])
    prediction = model.predict(vector)[0]
```

```
    return le.inverse_transform([prediction])[0]
```

18. 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)
```

```
Requirement already satisfied: gradio in /usr/local/lib/python3.11/dist-packages (5.29.1)
Requirement already satisfied: aiofiles<25.0,>=22.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (24.1.0)
```