```
# STEP 1: Upload CSV from local (Google Colab)
from google.colab import files
uploaded = files.upload()

# STEP 2: Load and clean dataset
import pandas as pd
import re

# Load the uploaded file (replace with the actual filename)
df = pd.read_csv("sentimentdataset.csv")

# Drop useless columns (if they exist)
df = df.drop(columns=["Unnamed: 0", "User"], errors='ignore')
df = df.dropna(subset=["Text", "Sentiment"])

# Clean and normalize sentiment labels
df["Sentiment"] = df["Sentiment"].astype(str).str.strip().str.lower()

# Optional: Merge sentiments into 3 main classes
positive_labels = ["joy", "happiness", "contentment", "excitement", "positive", "delight"]
neutral_labels = ["neutral", "calm", "okay", "fine"]
negative_labels = ["sad", "anger", "fear", "hate", "disgust", "negative", "worry", "depress

def map_sentiment(s):
    if s in positive_labels:
        return "positive"
    elif s in negative_labels:
        return "negative"
    elif s in neutral_labels:
        return "neutral"
    else:
        return None  # drop unknown classes

df["Sentiment"] = df["Sentiment"].apply(map_sentiment)
df = df.dropna(subset=["Sentiment"])  # Remove unknowns

# Text cleaning
def clean_text(text):
    text = re.sub(r"http\S+", "", text)
    text = re.sub(r"@\w+", "", text)
    text = re.sub(r"#\w+", "", text)
    text = re.sub(r"[^a-zA-Z\s]", "", text)
    return text.lower().strip()

df["Clean_Text"] = df["Text"].astype(str).apply(clean_text)

# STEP 3: Feature extraction with TF-IDF
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer(max_features=3000)
X = vectorizer.fit_transform(df["Clean_Text"])

# STEP 4: Encode sentiment labels
from sklearn.preprocessing import LabelEncoder

encoder = LabelEncoder()
y = encoder.fit_transform(df["Sentiment"])

# STEP 5: Split data & train RandomForest
```

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

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# STEP 6: Evaluate model
y_pred = model.predict(X_test)
print("Classification Report:")
print(classification_report(y_test, y_pred, target_names=encoder.classes_, zero_division=0)
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Classification Report:
              precision    recall  f1-score   support

    negative       1.00      0.25      0.40         4
     neutral       0.00      0.00      0.00         5
    positive       0.79      1.00      0.89        31

    accuracy                           0.80        40
   macro avg       0.60      0.42      0.43        40
weighted avg       0.72      0.80      0.73        40
```