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

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.
Saving sentimentdatasett.csv to sentimentdatasett.csv

```
import pandas as pd
import re
```

```
df = pd.read_csv("sentimentdatasett.csv")
```

```
print(df.head())
```

```
   Unnamed: 0.1  Unnamed: 0                                       Text  \
0             0           0       Enjoying a beautiful day at the park!
1             1           1          Traffic was terrible this morning.
2             2           2        Just finished an amazing workout! 馃捈
3             3           3  Excited about the upcoming weekend getaway!
4             4           4  Trying out a new recipe for dinner tonight.

  Sentiment            Timestamp        User   Platform            Hashtags  \
0  Positive  2023-01-15 12:30:00     User123    Twitter       #Nature #Park
1  Negative   2023-01-15 8:45:00   CommuterX    Twitter   #Traffic #Morning
2  Positive  2023-01-15 15:45:00  FitnessFan  Instagram   #Fitness #Workout
3  Positive  2023-01-15 18:20:00  AdventureX   Facebook  #Travel #Adventure
4   Neutral  2023-01-15 19:55:00    ChefCook  Instagram      #Cooking #Food

   Retweets  Likes    Country  Year  Month  Day  Hour
0        15     30        USA  2023      1   15    12
1         5     10     Canada  2023      1   15     8
2        20     40        USA  2023      1   15    15
3         8     15         UK  2023      1   15    18
4        12     25  Australia  2023      1   15    19
```

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

```
df["Sentiment"] = df["Sentiment"].astype(str).str.strip().str.lower()
```

```
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



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



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)


print(df[["Text", "Clean_Text"]].head())
```

```
                                          Text  \
0        Enjoying a beautiful day at the park!
1           Traffic was terrible this morning.
2         Just finished an amazing workout! 馃捈
3  Excited about the upcoming weekend getaway!
4  Trying out a new recipe for dinner tonight.

                                  Clean_Text
0        enjoying a beautiful day at the park
1           traffic was terrible this morning
2            just finished an amazing workout
3  excited about the upcoming weekend getaway
4  trying out a new recipe for dinner tonight
```

```
from sklearn.feature_extraction.text import TfidfVectorizer

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


from sklearn.preprocessing import LabelEncoder
```

```
encoder = LabelEncoder()

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)
```

```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

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

```
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
```