```
import numpy as np
import pandas as pd
import re
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import seaborn as sns
import matplotlib.pyplot as plt

# Download NLTK data
nltk.download('punkt')
nltk.download('stopwords')

# Load dataset
data_path = '/content/drive/MyDrive/emotion_data.csv'
data = pd.read_csv(data_path)

# Label Encoding
label_encoder = LabelEncoder()
data['emotion_label'] = label_encoder.fit_transform(data['emotion'])

# Text Cleaning Function
stop_words = set(stopwords.words('english'))

def clean_text(text):
    text = str(text).lower()
```

```
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    words = word_tokenize(text)
    words = [w for w in words if w not in stop_words]
    return ' '.join(words)

data['cleaned_text'] = data['text'].apply(clean_text)

# TF-IDF Vectorization
tfidf = TfidfVectorizer(max_features=3000)
X = tfidf.fit_transform(data['cleaned_text']).toarray()
y = data['emotion_label']

# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# ML Model (Logistic Regression)
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)

# Predictions
y_pred = model.predict(X_test)

# Evaluation
accuracy = accuracy_score(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)

print(f"Test Accuracy: {accuracy:.4f}")
print("\nClassification Report:\n", classification_report(y_test, y_pred))

# Confusion Matrix Plot
plt.figure(figsize=(8, 6))
```

```
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=label_encoder.classes_, yticklabels=label_encoder.classes_)
plt.title('Confusion Matrix - ML Model')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()

# Testing with new input text
test_texts = [
    "The festival in Saidpur was amazing today!",
    "I’m worried about the exam results."
]

test_data = pd.DataFrame({'text': test_texts})
test_data['cleaned_text'] = test_data['text'].apply(clean_text)
test_X = tfidf.transform(test_data['cleaned_text']).toarray()

predictions = model.predict(test_X)

for i, pred in enumerate(predictions):
    emotion = label_encoder.inverse_transform([pred])[0]
    print(f"Text {i+1}: {test_data['text'].iloc[i]}")
    print(f"Predicted Emotion: {emotion}\n")
```