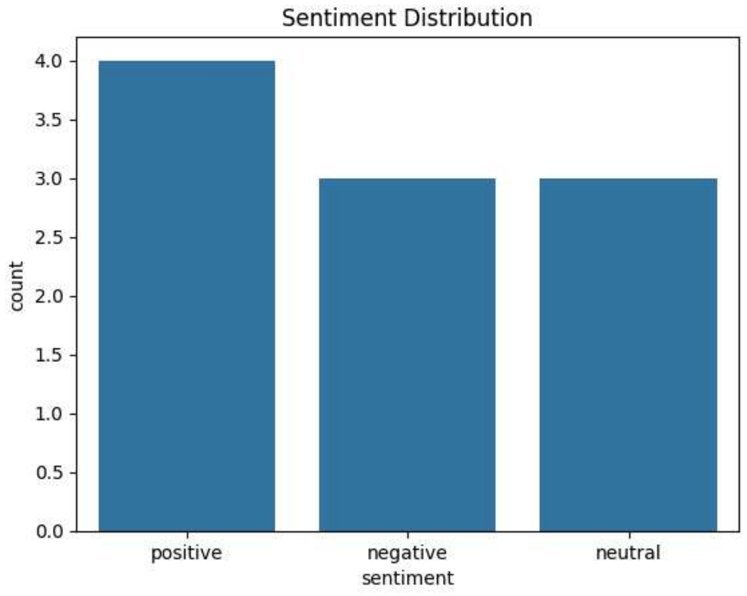
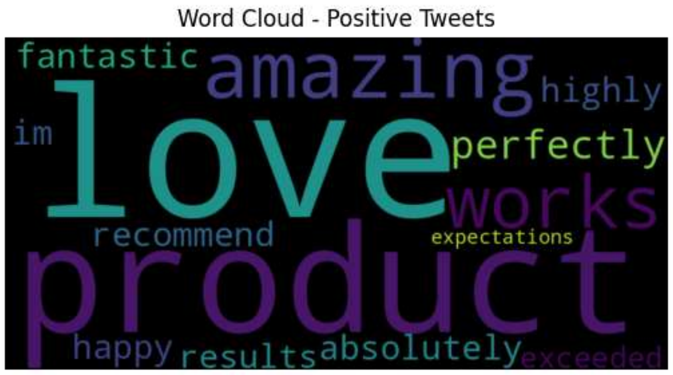
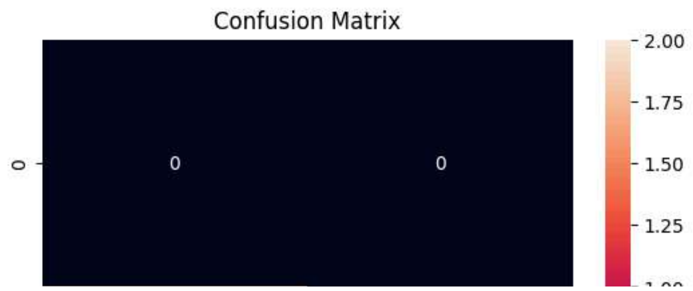
```
import pandas as pd
import numpy as np
import re
import string
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix

import nltk
from nltk.corpus import stopwords
from wordcloud import WordCloud

nltk.download('stopwords')

# Load data
df = pd.read_csv("/content/tweets.csv")  # Adjust filename
print(df.head())

# Clean tweets
def clean_text(text):
    text = re.sub(r"@\w+", "", text)  # Remove mentions
    text = re.sub(r"http\S+", "", text)  # Remove links
    text = re.sub(r"#", "", text)  # Remove hashtag symbol
    text = re.sub(r"[^\w\s]", "", text)  # Remove punctuation
    text = text.lower()  # Lowercase
    return text

df["clean_text"] = df["text"].apply(clean_text)

# Visualize sentiment distribution
sns.countplot(data=df, x="sentiment")
plt.title("Sentiment Distribution")
plt.show()

# WordCloud for positive tweets
pos_tweets = " ".join(df[df["sentiment"] == "positive"]["clean_text"])
wordcloud = WordCloud(width=800, height=400).generate(pos_tweets)
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.title("Word Cloud - Positive Tweets")
plt.show()

# Vectorize text
vectorizer = TfidfVectorizer(stop_words=stopwords.words("english"))
X = vectorizer.fit_transform(df["clean_text"])
y = df["sentiment"]

# Train-test split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2) # Removed stratify


# Model training
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)

# Evaluation
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt="d")
plt.title("Confusion Matrix")
plt.show()
```

```
                                                text sentiment
0  I love this product! It's amazing and works pe...  positive
1        This is the worst thing I have ever bought.  negative
2                         It’s okay, nothing special.   neutral
3         Absolutely fantastic! Highly recommend it.  positive
4                    I hate it. Terrible experience.  negative
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```





```
              precision    recall  f1-score   support

    negative       0.00      0.00      0.00       0.0
    positive       0.00      0.00      0.00       2.0

    accuracy                           0.00       2.0
   macro avg       0.00      0.00      0.00       2.0
weighted avg       0.00      0.00      0.00       2.0

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


Confusion Matrix