In [4]:
```
# 📌 1. IMPORT LIBRARIES
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import string
import nltk
from nltk.corpus import stopwords
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\manes\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
```

Out[4]:
```
True
```

In [5]:
```
# 📁 2. LOAD DATA
# Using a sample dataset of reviews. You can replace this with your own.
# We'll use the NLTK's movie_reviews corpus for simplicity.

from nltk.corpus import movie_reviews
import random

documents = [(movie_reviews.raw(fileid), category)
             for category in movie_reviews.categories()
             for fileid in movie_reviews.fileids(category)]

random.shuffle(documents)

# Convert into a DataFrame
data = pd.DataFrame(documents, columns=['review', 'sentiment'])
data.head()
```

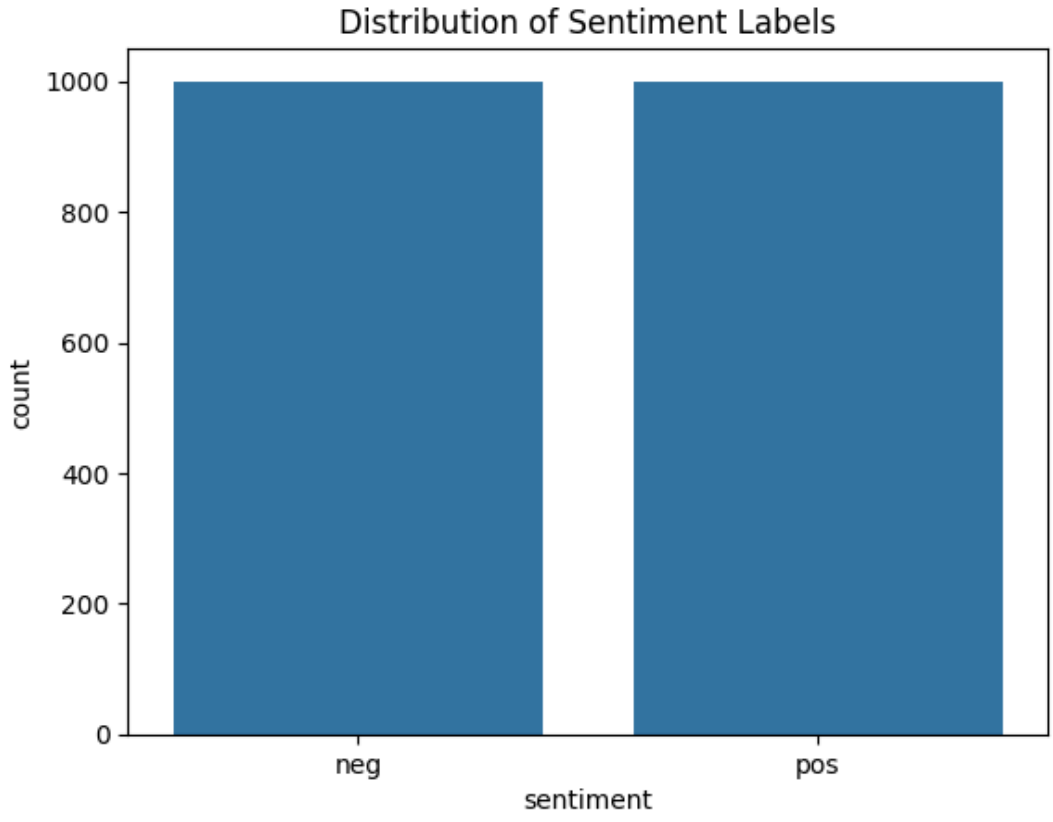
Out[5]:

	review	sentiment
0	what would you do if no one could see you ? \n...	neg
1	the " italian hitchcock " and acknowledged mas...	pos
2	senseless is a prime example of what can happe...	neg
3	the long kiss goodnight (r) meryl streep tri...	pos
4	2 days in the valley is more or less a pulp fi...	neg

In [6]:
```
# 📊 3. EXPLORE DATA
print("Total reviews:", len(data))
print(data['sentiment'].value_counts())

sns.countplot(x='sentiment', data=data)
plt.title("Distribution of Sentiment Labels")
plt.show()
```

```
Total reviews: 2000
sentiment
neg    1000
pos    1000
Name: count, dtype: int64
```



In [7]:
```
# 🧹 4. TEXT PREPROCESSING
stop_words = stopwords.words('english')

def clean_text(text):
    # Lowercase
    text = text.lower()
    # Remove punctuation
    text = ''.join([ch for ch in text if ch not in string.punctuation])
    # Remove stopwords
    tokens = text.split()
    tokens = [word for word in tokens if word not in stop_words]
    return ' '.join(tokens)

data['clean_review'] = data['review'].apply(clean_text)
data.head()
```

Out[7]:

	review	sentiment	clean_review
0	what would you do if no one could see you ? \n...	neg	would one could see well youre super smart bio...
1	the " italian hitchcock " and acknowledged mas...	pos	italian hitchcock acknowledged master giallo m...
2	senseless is a prime example of what can happe...	neg	senseless prime example happen try push onejok...
3	the long kiss goodnight (r) meryl streep tri...	pos	long kiss goodnight r meryl streep tried faile...
4	2 days in the valley is more or less a pulp fi...	neg	2 days valley less pulp fiction knock basicall...

In [8]:
```
# ✂️ 5. SPLIT DATA
X = data['clean_review']
y = data['sentiment']

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

In [9]:
```
# 🔠 6. VECTORIZATION (TEXT → NUMBERS)
vectorizer = TfidfVectorizer(max_features=5000)
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)
```

In [11]:
```
# 7. TRAIN MODEL
model = MultinomialNB()
model.fit(X_train_vec, y_train)

# ✅ 8. EVALUATION
y_pred = model.predict(X_test_vec)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

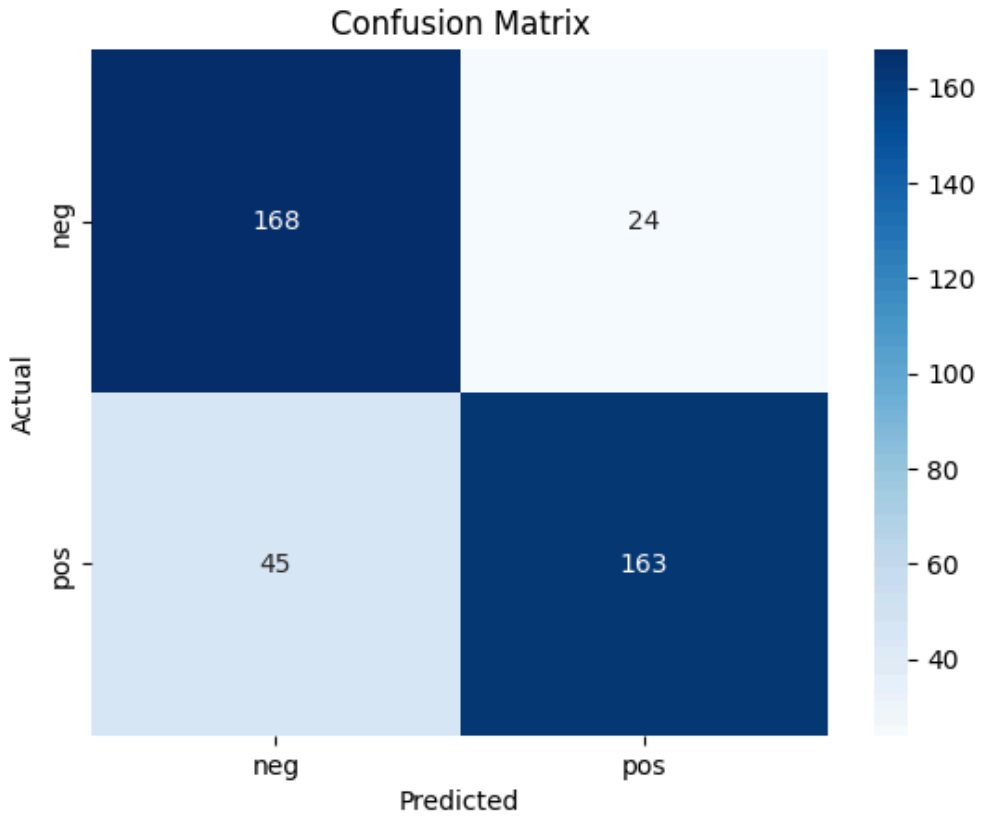
```
Accuracy: 0.8275

Classification Report:
              precision    recall  f1-score   support

         neg       0.79      0.88      0.83       192
         pos       0.87      0.78      0.83       208

    accuracy                           0.83       400
   macro avg       0.83      0.83      0.83       400
weighted avg       0.83      0.83      0.83       400
```

In [12]:
```
# 🌀 9. CONFUSION MATRIX
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['neg', 'pos'], yticklabe
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```



In [13]:
```
# 🎯 10. PREDICT FUNCTION
def predict_sentiment(review):
    cleaned = clean_text(review)
    vec = vectorizer.transform([cleaned])
    pred = model.predict(vec)[0]
    print(f"🗣️ Review: {review}\n🔮 Prediction: {pred.upper()}")

# 🔎 Try it!
predict_sentiment("This product is absolutely wonderful! Loved it.")
predict_sentiment("Worst experience ever. Waste of money.")
```

```
🗣️ Review: This product is absolutely wonderful! Loved it.
🔮 Prediction: POS
🗣️ Review: Worst experience ever. Waste of money.
🔮 Prediction: NEG
```