```
import nltk
from nltk.corpus import product_reviews_1, stopwords
from nltk.stem import PorterStemmer
from collections import defaultdict
import math
import random
```

0. Завантаження ресурсів

```
nltk.download('product_reviews_1')
nltk.download('stopwords')
```

```
[nltk_data] Downloading package product_reviews_1 to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package product_reviews_1 is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
True
```

1. Завантаження документів

```
documents = []

for fileid in product_reviews_1.fileids():
    # Визначаємо мітку: 1 - позитивний, 0 - негативний
    label = 1 if fileid.startswith('pos') else 0
    words = list(product_reviews_1.words(fileid))
    documents.append((words, label))

random.shuffle(documents)
split = int(0.8 * len(documents))
train_docs = documents[:split]
test_docs = documents[split:]

print(f"Розмір тренувальної вибірки: {len(train_docs)}, тестової: {len(test_docs)}")
```

```
Розмір тренувальної вибірки: 4, тестової: 2
```

2. Попередня обробка тексту

```
stemmer = PorterStemmer()
stop_words = set(stopwords.words('english'))

def process_tweet(words):
    """
    Токенізація, видалення стоп-слів і стемінг
    """
    return [stemmer.stem(w.lower()) for w in words if w.isalpha() and w.lower() not in stop_words]
```

3. Побудова словника частот

```
freqs = defaultdict(lambda: [0, 0])  # [негатив, позитив]

for words, label in train_docs:
```

```
    for w in process_tweet(words):
        freqs[w][label] += 1
```

4. Обчислення logprior з захистом від ділення на нуль

```
num_pos = sum(1 for _, y in train_docs if y == 1)
num_neg = sum(1 for _, y in train_docs if y == 0)

# Згладжування +1
num_pos += 1
num_neg += 1

logprior = math.log(num_pos / num_neg)
```

5. Обчислення loglikelihood з Laplace smoothing

```
V = len(freqs)
pos_words = sum(freqs[w][1] for w in freqs)
neg_words = sum(freqs[w][0] for w in freqs)

loglikelihood = {}
for w in freqs:
    p_w_pos = (freqs[w][1] + 1) / (pos_words + V)
    p_w_neg = (freqs[w][0] + 1) / (neg_words + V)
    loglikelihood[w] = math.log(p_w_pos / p_w_neg)
```

6. Наївний баєсів класифікатор

```
def naive_bayes_predict(words):
    score = logprior
    for w in process_tweet(words):
        if w in loglikelihood:
            score += loglikelihood[w]
    return 1 if score > 0 else 0
```

7. Оцінка точності

```
correct = 0
for words, label in test_docs:
    if naive_bayes_predict(words) == label:
        correct += 1

accuracy = correct / len(test_docs)
print(f"\nТочність класифікатора: {accuracy*100:.2f}%")
```

```
Точність класифікатора: 100.00%
```

8. Найбільш позитивні та негативні слова

```
sorted_words = sorted(loglikelihood.items(), key=lambda x: x[1])
print("\nТоп 10 негативних слів:")
```

```
for w, s in sorted_words[:10]:
    print(w, round(s,3))

print("\nТоп 10 позитивних слів:")
for w, s in sorted_words[-10:]:
    print(w, round(s,3))
```

```
Топ 10 негативних слів:
player -3.885
use -3.407
phone -3.407
n -3.392
one -3.238
dvd -3.098
get -2.998
play -2.958
would -2.929
ipod -2.929

Топ 10 позитивних слів:
seach 1.502
cheer 1.502
satisfactorili 1.502
worldphon 1.502
italian 1.502
german 1.502
dutch 1.502
browser 1.502
fed 1.502
geek 1.502
```

9. Тест на власному відгуку

```
my_review = "This product is amazing and very useful"
prediction = naive_bayes_predict(my_review.split())
print("\nМій відгук:", my_review)
print("Прогноз:", "Позитивний" if prediction==1 else "Негативний")
```

```
Мій відгук: This product is amazing and very useful
Прогноз: Негативний
```