Лабораторна робота № 1 "Застосування логістичної регресії для аналізу тональності текстів"

```
import nltk
import numpy as np
import re
import string
from nltk.corpus import stopwords, twitter_samples
from nltk.stem import PorterStemmer
from nltk.tokenize import TweetTokenizer
```

ЗАВДАННЯ 2: Функції попередньої обробки

```
def process_tweet(tweet):
    """
    Обробляє твіт: токенізація, видалення стоп-слів, стемінг.
    """
    stemmer = PorterStemmer()
    stopwords_english = stopwords.words('english')

    # Видалення $GE, RT, посилань
    tweet = re.sub(r'\$\w*', '', tweet)
    tweet = re.sub(r'^RT[\s]+', '', tweet)
    tweet = re.sub(r'https?://[^\s\n\r]+', '', tweet)
    # Видалення хештегів (#)
    tweet = re.sub(r'#', '', tweet)

    # Токенізація
    tokenizer = TweetTokenizer(preserve_case=False, strip_handles=True, reduce_len=True)
    tweet_tokens = tokenizer.tokenize(tweet)

    tweets_clean = []
    for word in tweet_tokens:
        if (word not in stopwords_english and  # видалення стоп-слів
            word not in string.punctuation):   # видалення пунктуації
            stem_word = stemmer.stem(word)  # стемінг
            tweets_clean.append(stem_word)

    return tweets_clean
```

ЗАВДАННЯ 3: Побудова словника частотності

```
def build_freqs(tweets, ys):
    """
    Будує словник частот freqs[(word, label)] = count.
    """
    # Конвертуємо np array в list
    yslist = np.squeeze(ys).tolist()

    freqs = {}
    for y, tweet in zip(yslist, tweets):
        for word in process_tweet(tweet):
            pair = (word, y)
            freqs[pair] = freqs.get(pair, 0) + 1

    return freqs
```

ЗАВДАННЯ 4: Реалізація логістичної регресії

```
def sigmoid(z):
    """Сигмоїдна функція"""
    h = 1 / (1 + np.exp(-z))
    return h

def gradientDescent(x, y, theta, alpha, num_iters):
    """
    Реалізація градієнтного спуску.
    """
    m = x.shape[0]

    # Переконуємося, що y має правильну форму (m, 1)
    if y.ndim == 1:
        y = y.reshape(-1, 1)

    for i in range(0, num_iters):
        # Обчислюємо гіпотезу
```

```
            z = np.dot(x, theta)
            h = sigmoid(z)

            # Обчислюємо функцію втрат (log loss)
            epsilon = 1e-15  # Маленька константа для уникнення log(0)
            h_safe = np.clip(h, epsilon, 1 - epsilon) # Обмежуємо значення
            J = (-1/m) * np.sum(y * np.log(h_safe) + (1-y) * np.log(1 - h_safe))

            # Обчислюємо градієнт
            grad = (1/m) * np.dot(x.T, (h - y))

            # Оновлюємо параметри
            theta = theta - alpha * grad

            if i % 100 == 0: # Друкуємо лог кожні 100 ітерацій
                print(f"Ітерація {i}, Функція втрат J = {J:.8f}")

        return J, theta
```

ЗАВДАННЯ 5: Функції вилучення ознак та передбачення

```
def extract_features(tweet, freqs):
    """
    Формує вектор ознак (1, 3) для одного твіту.
    Ознаки: [bias, sum(pos_freqs), sum(neg_freqs)]
    """
    word_l = process_tweet(tweet)
    x = np.zeros((1, 3))
    x[0,0] = 1  # Bias

    for word in word_l:
        # Додаємо частоту слова з позитивного класу
        x[0,1] += freqs.get((word, 1.0), 0)
        # Додаємо частоту слова з негативного класу
        x[0,2] += freqs.get((word, 0.0), 0)

    assert(x.shape == (1, 3))
    return x
```

```
def predict_tweet(tweet, freqs, theta):
    """
    Передбачає тональність одного твіту.
    """
    x = extract_features(tweet, freqs)
    y_pred = sigmoid(np.dot(x, theta))
    return y_pred
```

```
def test_logistic_regression(test_x, test_y, freqs, theta):
    """
    Тестує модель на тестовій вибірці та повертає точність.
    """
    y_hat = []

    for tweet in test_x:
        y_pred = predict_tweet(tweet, freqs, theta)

        if y_pred > 0.5:
            y_hat.append(1.0)
        else:
            y_hat.append(0.0)

    # Порівнюємо прогнози (y_hat) з реальними мітками (test_y)
    # np.squeeze() для коректного порівняння форм
    accuracy = np.mean(np.array(y_hat) == np.squeeze(test_y))
    return accuracy
```

ГОЛОВНИЙ БЛОК ВИКОНАННЯ

```
if __name__ == "__main__":

    # 1. Завантаження NLTK даних
    nltk.download('twitter_samples', quiet=True)
    nltk.download('stopwords', quiet=True)
    nltk.download('punkt', quiet=True) # Потрібен для TweetTokenizer

    # 2. (Завдання 1 & 2) Завантаження та розділення корпусу (як у описі)
    print("Завантаження та розділення корпусу 'twitter_samples'...")
    all_positive_tweets = twitter_samples.strings('positive_tweets.json')
```

```
all_negative_tweets = twitter_samples.strings('negative_tweets.json')

# 4000 на навчання, 1000 на тест
test_pos = all_positive_tweets[4000:]
train_pos = all_positive_tweets[:4000]
test_neg = all_negative_tweets[4000:]
train_neg = all_negative_tweets[:4000]

train_x = train_pos + train_neg
test_x = test_pos + test_neg

# Створюємо мітки: 1.0 для позитивних, 0.0 для негативних
train_y = np.append(np.ones(len(train_pos)), np.zeros(len(train_neg)))
test_y = np.append(np.ones(len(test_pos)), np.zeros(len(test_neg)))
print("Дані підготовлено.")

# 3. (Завдання 3) Побудова словника частотності
print("\nПобудова словника частот...")
freqs = build_freqs(train_x, train_y)
print(f"Словник частот побудовано. {len(freqs)} унікальних пар.")

# 4. (Завдання 4 & 5) Навчання моделі
print("\n--- Початок навчання моделі ---")

# Створення матриці ознак X (m, 3)
X = np.zeros((len(train_x), 3))
for i in range(len(train_x)):
    X[i, :]= extract_features(train_x[i], freqs)

Y = train_y # Y - це train_y

# Гіперпараметри
alpha = 1e-9
num_iters = 1500

# Навчання
J, theta = gradientDescent(X, Y, np.zeros((3, 1)), alpha, num_iters)

print("\n--- Навчання завершено ---")
print(f"Фінальна помилка (Loss): {J:.8f}")
print(f"Фінальні ваги (Theta): {[round(t, 8) for t in np.squeeze(theta)]}")

# 5. (Завдання 6) Оцінка точності
accuracy = test_logistic_regression(test_x, test_y, freqs, theta)
print(f"\nТочність логістичної регресії на тестовій вибірці: {accuracy * 100:.2f}%")

# 6. (Завдання 6 & 7) Тестування на власних прикладах
print("\n--- Тестування на власних прикладах ---")
my_tweet_pos = 'I am so happy and excited, this is the best day ever!'
y_hat_pos = predict_tweet(my_tweet_pos, freqs, theta)
print(f"Твіт: '{my_tweet_pos}'")
print(f"Прогноз: {y_hat_pos[0][0]:.4f} -> {'Позитивний' if y_hat_pos > 0.5 else 'Негативний'}")

my_tweet_neg = 'This is a terrible and awful experience. I hate it.'
y_hat_neg = predict_tweet(my_tweet_neg, freqs, theta)
print(f"\nТвіт: '{my_tweet_neg}'")
```

```
Завантаження та розділення корпусу 'twitter_samples'...
Дані підготовлено.

Побудова словника частот...
Словник частот побудовано. 11397 унікальних пар.

--- Початок навчання моделі ---
Ітерація 0, Функція втрат J = 0.69314718
Ітерація 100, Функція втрат J = 0.59538303
Ітерація 200, Функція втрат J = 0.52206432
Ітерація 300, Функція втрат J = 0.46560367
Ітерація 400, Функція втрат J = 0.42105697
Ітерація 500, Функція втрат J = 0.38517211
Ітерація 600, Функція втрат J = 0.35574646
Ітерація 700, Функція втрат J = 0.33124495
Ітерація 800, Функція втрат J = 0.31057008
Ітерація 900, Функція втрат J = 0.29291936
Ітерація 1000, Функція втрат J = 0.27769420
Ітерація 1100, Функція втрат J = 0.26444028
Ітерація 1200, Функція втрат J = 0.25280730
Ітерація 1300, Функція втрат J = 0.24252135
Ітерація 1400, Функція втрат J = 0.23336560

--- Навчання завершено ---
Фінальна помилка (Loss): 0.22524410
Фінальні ваги (Theta): [np.float64(6e-08), np.float64(0.00053786), np.float64(-0.00055885)]

Точність логістичної регресії на тестовій вибірці: 99.65%
```

```
--- Тестування на власних прикладах ---
Твіт: 'I am so happy and excited, this is the best day ever!'
Прогноз: 0.5349 -> Позитивний

Твіт: 'This is a terrible and awful experience. I hate it.'
```