Đề 10

**Bài tập 10: Phân tích cảm xúc trên đánh giá sản phẩm**

**Mô tả:** Sử dụng tập dữ liệu đánh giá sản phẩm để phân loại cảm xúc thành tích cực và tiêu cực.

**Yêu cầu:**

1. Tải tập dữ liệu và chia thành tập huấn luyện (80%) và tập kiểm tra (20%).

2. Thực hiện tiền xử lý văn bản (xóa dấu câu, mã hóa văn bản).

3. Sử dụng phương pháp TF-IDF để chuyển đổi văn bản thành dạng số.

4. Áp dụng thuật toán Naive Bayes để xây dựng mô hình phân loại.

5. Đánh giá mô hình bằng độ chính xác và F1-score.

6. Đề xuất phương pháp cải thiện dự đoán.

Import thư viện

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.naive\_bayes import MultinomialNB

from sklearn.metrics import accuracy\_score, f1\_score

import string

import re

+ Tải tệp dữ liệu

# 'review' chứa văn bản đánh giá sản phẩm, 'sentiment' là nhãn (1: tích cực, 0: tiêu cực)

data = pd.read\_csv('product\_reviews.csv')

+ Chia tệp dữ liệu

X = data['review']  # Đánh giá sản phẩm

y = data['sentiment']  # Nhãn cảm xúc (1: tích cực, 0: tiêu cực)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

1. Tiền xử lý văn bản

def preprocess\_text(text):

    # Chuyển thành chữ thường

    text = text.lower()

    # Xóa dấu câu

    text = text.translate(str.maketrans('', '', string.punctuation))

    # Xóa số và các ký tự không mong muốn

    text = re.sub(r'\d+', '', text)

    return text

# Áp dụng hàm tiền xử lý cho dữ liệu huấn luyện và kiểm tra

X\_train = X\_train.apply(preprocess\_text)

X\_test = X\_test.apply(preprocess\_text)

1. Sử dụng phương pháp TF-IDF để chuyển đổi văn bản thành dạng số.

tfidf = TfidfVectorizer(max\_features=5000)  # Lấy 5000 từ quan trọng nhất

X\_train\_tfidf = tfidf.fit\_transform(X\_train)

X\_test\_tfidf = tfidf.transform(X\_test)

1. Áp dụng thuật toán Naive Bayes để xây dựng mô hình phân loại.

nb\_model = MultinomialNB()

nb\_model.fit(X\_train\_tfidf, y\_train)

1. Đánh giá mô hình bằng độ chính xác và F1-score.

y\_pred = nb\_model.predict(X\_test\_tfidf)

# Độ chính xác

accuracy = accuracy\_score(y\_test, y\_pred)

# F1-score

f1 = f1\_score(y\_test, y\_pred)

1. Đề xuất phương pháp cải thiện dự đoán.

\* Sử dụng các thuật toán phức tạp hơn

from sklearn.svm import SVC

from sklearn.ensemble import RandomForestClassifier

from sklearn.linear\_model import LogisticRegression

# Thử nghiệm với Logistic Regression

lr\_model = LogisticRegression(max\_iter=1000)

lr\_model.fit(X\_train\_tfidf, y\_train)

y\_pred\_lr = lr\_model.predict(X\_test\_tfidf)

# Đánh giá Logistic Regression

accuracy\_lr = accuracy\_score(y\_test, y\_pred\_lr)

f1\_lr = f1\_score(y\_test, y\_pred\_lr)

print(f"Độ chính xác của Logistic Regression: {accuracy\_lr:.2f}")

print(f"F1-score của Logistic Regression: {f1\_lr:.2f}")

\* Sử dụng GridSearchCV để tìm tham số tối ưu cho Naive Bayes

from sklearn.model\_selection import GridSearchCV

# Thử nghiệm với Multinomial Naive Bayes và GridSearchCV

param\_grid = {'alpha': [0.1, 0.5, 1.0, 2.0]}  # Thử nghiệm các giá trị alpha khác nhau

grid\_search = GridSearchCV(MultinomialNB(), param\_grid, cv=5, scoring='accuracy')

grid\_search.fit(X\_train\_tfidf, y\_train)

# Tìm alpha tốt nhất

best\_nb\_model = grid\_search.best\_estimator\_

y\_pred\_best\_nb = best\_nb\_model.predict(X\_test\_tfidf)

# Đánh giá mô hình tốt nhất

accuracy\_best\_nb = accuracy\_score(y\_test, y\_pred\_best\_nb)

f1\_best\_nb = f1\_score(y\_test, y\_pred\_best\_nb)

print(f"Độ chính xác của Naive Bayes tốt nhất (GridSearchCV): {accuracy\_best\_nb:.2f}")

print(f"F1-score của Naive Bayes tốt nhất (GridSearchCV): {f1\_best\_nb:.2f}")