Spam Detection with Logistic Regression

Tên: Nguyễn Gia Huy
MSSV:2251320012

Bước 1. Nhập các thư viện cần thiết

- **NumPy**: Hỗ trợ tính toán ma trận và toán học số học.
- **Pandas**: Xử lý và phân tích dữ liệu.
- **Scikit-learn**:

  - train_test_split: Chia dữ liệu thành tập huấn luyện và kiểm thử.

  - TfidfVectorizer: Biến đổi văn bản thành các đặc trưng số (TF-IDF).

  - accuracy_score: Đánh giá độ chính xác của mô hình.

- **SciPy**: Dùng để xử lý ma trận thưa.

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import accuracy_score
from scipy.sparse import csr_matrix
```

Bước 2. Lớp Logistic Regression

Tạo mô hình hồi quy logistic từ đầu, với các chức năng chính:

1. **sigmoid(z):**

   - Hàm kích hoạt Sigmoid, chuẩn hóa kết quả thành khoảng [0, 1].

2. **compute_loss(y_pred, y):**

   - Tính toán hàm mất mát (binary cross-entropy) để đánh giá sự khác biệt giữa dự đoán và nhãn thực.

3. **fit(X, y):**

☐ Huấn luyện mô hình bằng Gradient Descent:

. Cập nhật trọng số (weights) và độ chệch (bias) để giảm hàm mất mát.

☐ X là dữ liệu đầu vào, y là nhãn mục tiêu.

4. **predict(X)**:

☐ Dự đoán nhãn cho dữ liệu mới:

.● Tính xác suất thông qua Sigmoid.

● Gán nhãn 1 (Ham) nếu xác suất > 0.5, ngược lại gán nhãn 0 (Spam).

```
class LogisticRegression:
    def __init__(self, learning_rate=0.01, num_iterations=1000):
        self.learning_rate = learning_rate
        self.num_iterations = num_iterations
        self.weights = None
        self.bias = None
        self.loss_history = []

    def sigmoid(self, z):
        return 1 / (1 + np.exp(-z))

    def compute_loss(self, y_pred, y):
        epsilon = 1e-10
        loss = -np.mean(y * np.log(y_pred + epsilon) + (1 - y) * np.log(1 - y_pred + epsilon))
        return loss

    def fit(self, X, y):
        num_samples, num_features = X.shape
        if isinstance(X, csr_matrix):
            X = X.toarray()
        self.weights = np.zeros(num_features)
        self.bias = 0
        for _ in range(self.num_iterations):
            linear_model = np.dot(X, self.weights) + self.bias
            y_pred = self.sigmoid(linear_model)
            loss = self.compute_loss(y_pred, y)
            self.loss_history.append(loss)
            dw = (1 / num_samples) * np.dot(X.T, (y_pred - y))
            db = (1 / num_samples) * np.sum(y_pred - y)
            self.weights -= self.learning_rate * dw
            self.bias -= self.learning_rate * db

    def predict(self, X):
        if isinstance(X, csr_matrix):
            X = X.toarray()
        linear_model = np.dot(X, self.weights) + self.bias
        y_pred = self.sigmoid(linear_model)
        y_pred_class = np.where(y_pred > 0.5, 1, 0)
        return y_pred_class
```

Bước 3. Đọc và tiền xử lý dữ liệu

```
raw_mail_data = pd.read_csv('mail_SPAM_data.csv')

mail_data = raw_mail_data.where((pd.notnull(raw_mail_data)), '')
```

- pd.read_csv: Đọc dữ liệu từ file CSV (mail_data.csv).
- pd.notnull: Loại bỏ giá trị null và thay thế bằng chuỗi rỗng.

Lưu ý phải thay path đường dẫn file .csv mới chạy được!

◎ **Mã hóa nhãn**:

```
mail_data.loc[mail_data['Category'] == 'spam', 'Category'] = 0
mail_data.loc[mail_data['Category'] == 'ham', 'Category'] = 1
```

- Gán nhãn spam thành **0** và ham thành **1**

◎ **Tách dữ liệu đầu vào và nhãn mục tiêu**:

```
X = mail_data['Message']
Y = mail_data['Category'].astype(int)
```

· X: Cột chứa nội dung tin nhắn.
· Y: Cột chứa nhãn (spam hoặc ham).

Bước 4. Chia tập dữ liệu và tạo đặc trưng

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=5)
```

Chia dữ liệu thành:

- **80%** để huấn luyện.
- **20%** để kiểm thử.

- random_state: Đảm bảo kết quả chia dữ liệu không thay đổi khi chạy lại mã.

Tạo đặc trưng văn bản bằng TF-IDF:

```
feature_extraction = TfidfVectorizer(min_df=1, stop_words='english', lowercase=True)
X_train_feature = feature_extraction.fit_transform(X_train)
X_test_features = feature_extraction.transform(X_test)
```

TfidfVectorizer:

- Chuyển đổi văn bản thành ma trận trọng số TF-IDF.
- stop_words='english': Loại bỏ các từ dừng phổ biến trong tiếng Anh (như "the", "is").
- lowercase=True: Chuyển tất cả văn bản thành chữ thường.

Bước 5. Huấn luyện mô hình

```
model = LogisticRegression()
model.fit(X_train_feature, Y_train)
```

- Khởi tạo mô hình Logistic Regression.
- Huấn luyện trên dữ liệu đã xử lý.

Bước 6. Dự đoán dữ liệu mới

Nhập đường dẫn file CSV cần dự đoán:

```
file_name = input("Nhập path file CSV chứa dữ liệu cần dự đoán: ")

new_mail_data = pd.read_csv(file_name)
```

- Đọc file chứa tin nhắn cần dự đoán.

Tạo đặc trưng cho dữ liệu mới:

```
new_messages = new_mail_data['Message']
new_messages_features = feature_extraction.transform(new_messages)
```

- Chuyển đổi văn bản thành ma trận TF-IDF.

Dự đoán và gán nhãn:

```
new_predictions = model.predict(new_messages_features)

new_mail_data['Prediction'] = new_predictions
new_mail_data['Prediction'] = new_mail_data['Prediction'].map({1: 'Ham', 0: 'Spam'})
```

- Sử dụng mô hình để dự đoán nhãn (Ham hoặc Spam).
- Gán nhãn dự đoán vào cột mới trong DataFrame.

Hiển thị kết quả:

```
print(new_mail_data[['Message', 'Prediction']])
```

- Hiển thị nội dung tin nhắn và nhãn dự đoán tương ứng

Source code github: https://github.com/HUyEsona/-ML-project_-Classifying-Spam-Emails.git