Comprehensive Documentation for URL Analysis Application

Overview

This application provides a machine learning-based URL analysis service that classifies URLs as malicious or benign. It includes user registration, authentication, feedback mechanisms, and model management features through a RESTful API built using Flask.

Components

1. Model Training (model.py)

This script is responsible for loading data, training a logistic regression model, and saving the model and associated components for later use.

Key Features

- **Data Loading**: Reads a CSV file containing URLs and their corresponding labels.
- **Label Encoding**: Encodes categorical labels to numerical format.
- **Data Splitting**: Divides the dataset into training and testing sets.
- **Text Vectorization**: Converts URLs into TF-IDF feature vectors.
- **Model Training**: Trains a logistic regression model on the processed data.
- **Model Evaluation**: Outputs accuracy, classification report, and confusion matrix.
- **Model Saving**: Saves the trained model, vectorizer, and training data to disk for later use.

2. Flask Application (app.py)

This script implements the Flask web application, providing endpoints for user interaction, URL analysis, and model management.

Key Features

- **User Registration and Login**: Secure user management using hashed passwords and JWT for authentication.
- **URL Analysis**: Allows authenticated users to analyze URLs and receive maliciousness scores.
- **Feedback System**: Collects user feedback and retrains the model based on sufficient feedback.
- **Data Management**: Provides endpoints for retrieving and deleting users and URL analyses.
- **Model Uploading**: Allows administrators to upload new models and associated data to update the application.

API Endpoints

1. User Management

Register a New User

- Endpoint: `/api/register`
- Method: `POST`
- Request Body:
  ```json
  {
    "username": "your_username",
    "email": "your_email@example.com",
    "password": "your_password"
  }
  ```
- Response:
  ```json
  {
    "status": "success",
    "userId": 1
  }
  ```

User Login

- Endpoint: `/api/login`

- Method: `POST`
- Request Body:
  ```json
  {
    "username": "your_username",
    "password": "your_password"
  }
  ```
- Response:
  ```json
  {
    "status": "success",
    "token": "your_jwt_token"
  }
  ```
- Error Response (Invalid Credentials):
  ```json
  {
    "status": "failure",
    "message": "Invalid credentials"
  }
  ```

User Logout

- Endpoint: `/api/logout`
- Method: `POST`
- Headers:
  ```
  Authorization: Bearer your_jwt_token
  ```
- Response:
  ```json
  {
    "status": "success",
    "message": "Logged out"
  }
  ```

2. URL Analysis

Analyze a URL

- Endpoint: `/api/analyze`
- Method: `POST`
- Headers:
  ```
  Authorization: Bearer your_jwt_token
  ```
- Request Body:
  ```json
  {
    "url": "http://example.com"
  }
  ```
- Response:
  ```json
  {
    "url": "http://example.com",
    "maliciousness_score": 0.95
  }
  ```

3. Feedback Submission

Submit Feedback

- Endpoint: `/api/feedback`
- Method: `POST`
- Headers:
  ```
  Authorization: Bearer your_jwt_token
  ```
- Request Body:
  ```json
  {
    "url": "http://example.com",
    "label": 1  // 1 for malicious, 0 for benign
  }
  ```
- Response:
  ```json
  {
    "status": "success"
  }
  ```

4. Data Retrieval

Get All Users

- Endpoint: `/api/users`
- Method: `GET`
- Headers:
  ```
  Authorization: Bearer your_jwt_token
  ```
- Response:
  ```json
  {
    "users": [
      {
        "id": 1,
        "username": "user1",
        "email": "user1@example.com"
      }
    ]
  }
  ```

Get All URL Analyses

- Endpoint: `/api/urls`
- Method: `GET`
- Headers:
  ```
  Authorization: Bearer your_jwt_token
  ```
- Response:
  ```json
  {
    "urls": [
      {
        "id": 1,
        "url": "http://example.com",
        "maliciousness_score": 0.95
      }
    ]
  }
```

```

## 5. **Data Deletion**

**Delete a User**

- Endpoint: `/api/user/<int:user_id>`
- Method: `DELETE`
- Headers:
  ```
  Authorization: Bearer your_jwt_token
  ```
- Response:
  ```json
  {
    "status": "success",
    "message": "User and all associated URLs deleted"
  }
  ```

Delete a URL Analysis

- Endpoint: `/api/url/<int:url_id>`
- Method: `DELETE`
- Headers:
  ```
  Authorization: Bearer your_jwt_token
  ```
- Response:
  ```json
  {
    "status": "success",
    "message": "URL analysis deleted"
  }
  ```

6. Model Management

Upload New Model

- Endpoint: `/api/upload_model`
  ```

- Method: `POST`
- Headers:
  ```
  Authorization: Bearer your_jwt_token
  ```
- Request Body: (Multipart/form-data)
  - model: `lr_model.pkl`
  - vectorizer: `vectorizer.pkl`
  - training_data: `training_data.pkl`
- Response:
  ```json
  {
    "status": "success",
    "message": "Model updated successfully"
  }
  ```

Retraining the Model

The model will be retrained automatically when the feedback count reaches 10. The `retrain_model` function will:

1. Retrieve all feedback.
2. Combine feedback with existing training data.
3. Train the model on the combined dataset.
4. Save the updated model to disk.

Conclusion

This application serves as a comprehensive framework for URL analysis, leveraging machine learning and providing secure user authentication. You can extend the functionality by adding features such as logging, monitoring, and more sophisticated model retraining strategies.