Project Title: Sentiment Analysis API

Project Description:

I build a backend service that exposes a RESTful API endpoint for sentiment analysis using Flask. The API will accept text input and return the sentiment analysis result using a pre-trained machine learning model.

Project setup:

1. Install Flask and other dependencies:
   - Create a new directory for this project and navigate to it.
   - Set up a virtual environment (optional but recommended):

     python -m venv env
   - Activate the virtual environment:

     venv\Scripts\activate.ps1
   - Install Flask: **pip install flask**
   - To use this model for inference, first install the SetFit library:

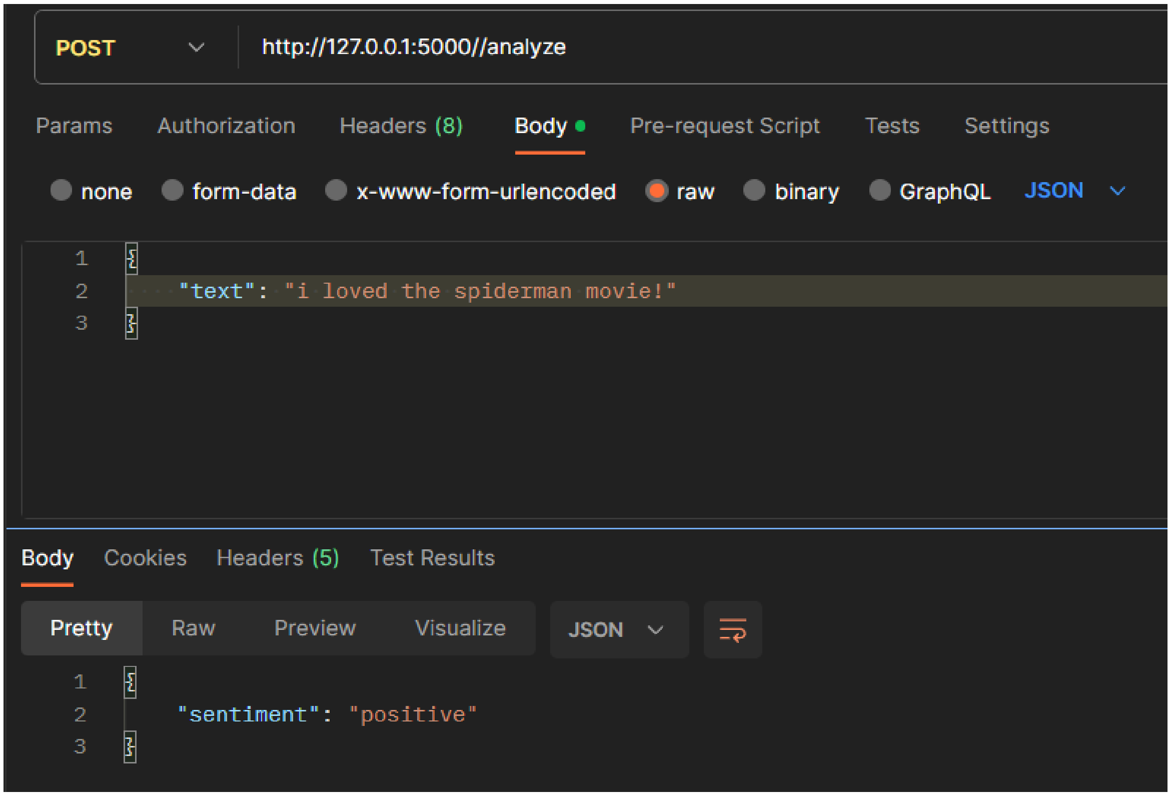
     python -m pip install setfit

2. Create a new file named **app.py** in this project directory. This file will contain the code for this Flask application.

   Implementing the API:
   Open app.py in a text editor and write the following code:

```
from flask import Flask, jsonify, request
from setfit import SetFitModel

app = Flask(__name__)

@app.route('/analyze', methods=['POST'])
def analyze_sentiment():
    data = request.get_json()

    # To Handle Error like "text" data not exits in JSON object.
    if 'text' not in data:
        return jsonify({'error': 'Invalid request. Missing "text" parameter.'})

    text = data['text']

    # Download from Hub and run inference

    model = SetFitModel.from_pretrained("StatsGary/setfit-ft-sentinent-eval")
    # Perform sentiment analysis on the text
    sentiment = model([text])

    #To handle Error
    # raise TypeError(f"Object of type {type(o).__name__} is not JSON
serializable")
    # TypeError: Object of type Tensor is not JSON serializable
    sentiment = sentiment.tolist()

    if sentiment[0] == 1:
        sentiment = 'positive'
    elif sentiment[0] == 0:
        sentiment = 'negative'
    else:
        sentiment = 'neutral'


    # The sentiment value is assigned to the "sentiment" key in the respon
dictionary

    respon = {
        'sentiment': sentiment
    }

    return jsonify(respon)


if __name__ == '__main__':
    app.run(debug=True)
```

3. Using postman for testing and interacting with the API endpoints by sending HTTP requests and receiving responses.



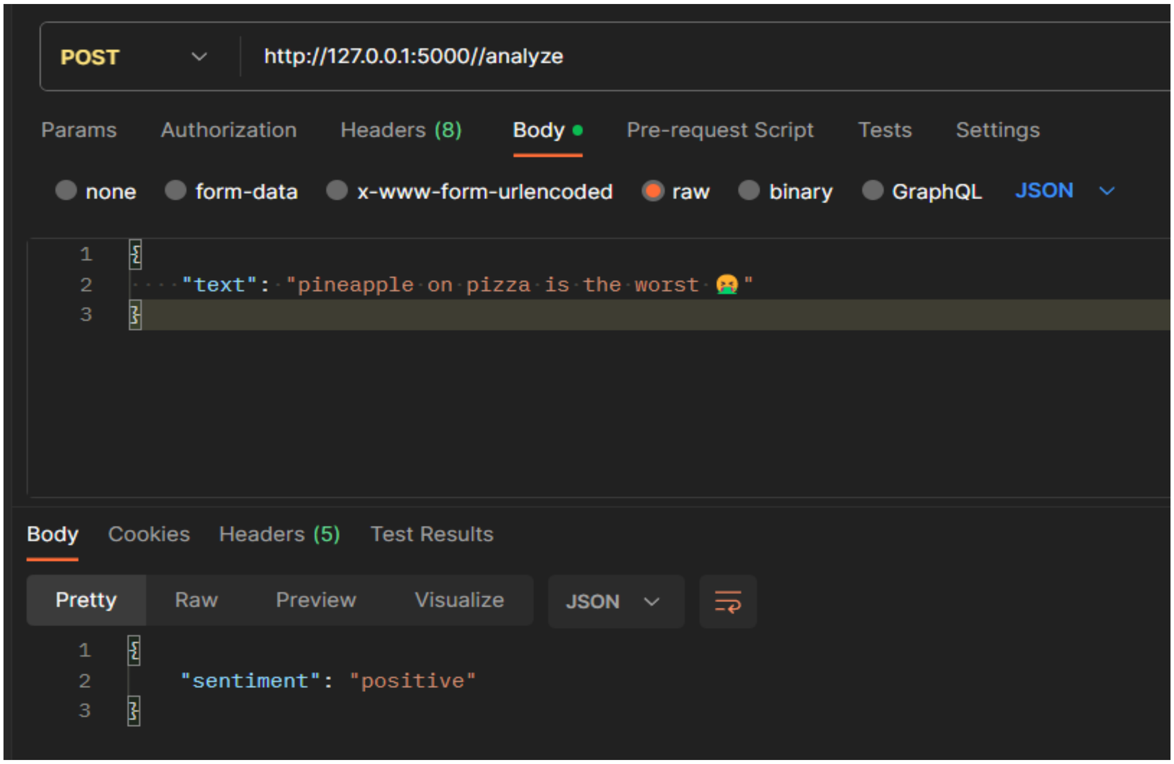
POST http://127.0.0.1:5000//analyze

Body: ○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```
{
    "text": "i loved the spiderman movie!"
}
```

Body — Pretty — JSON

```
{
    "sentiment": "positive"
}
```



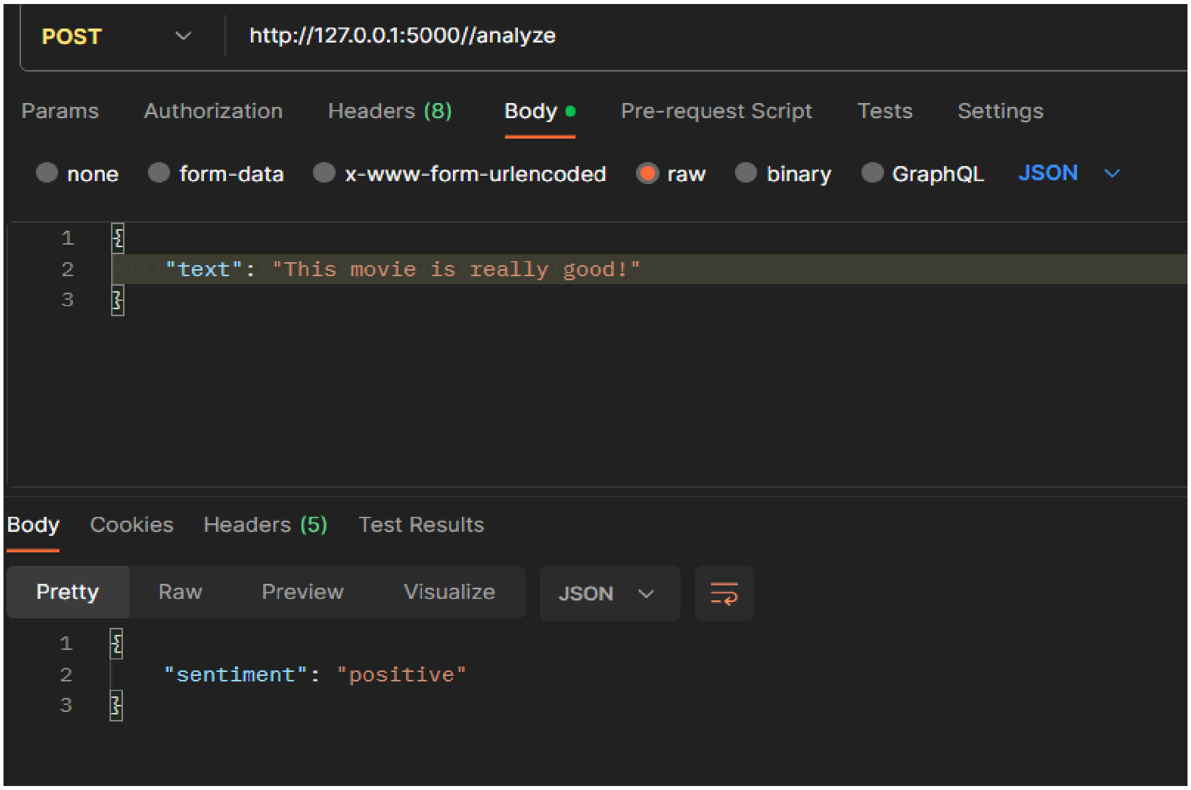
POST http://127.0.0.1:5000//analyze

Body: ○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```
{
    "text": "pineapple on pizza is the worst 🤮"
}
```

Body — Pretty — JSON

```
{
    "sentiment": "positive"
}
```



POST http://127.0.0.1:5000//analyze

Body: ● raw JSON

```
{
    "text": "This movie is really good!"
}
```

Body: Pretty JSON

```
{
    "sentiment": "positive"
}
```