Volumetric Analysis with Coordinate Transformations

Presented By (BDS-2B):

Muhammad Soban Sohail (24L-2545)

Abdul Muiz (24L-2501)

Adnan Zaka (24i-2537)

Mahad Jawad (24L-2504)

Rai Fahad Sultan (24L-2509)

Taimur Amir (24L-2518)

Saad Jahangir (24L-2516)

Umer Khan (24L-2583)

Abdullah Zia Chaudhary (24L-2507)

Submitted To:

Mr. Muhammad Yaseen

Table of Contents

1. Introduction

This project is a Flask-based web application that predicts the optimal coordinate system (Cartesian, Cylindrical, or Spherical) for solving integration problems. It extracts mathematical equations from user queries, visualizes them using 2D/3D plotting, computes the volume integrals symbolically using SymPy, and renders the results interactively on a web page.

2. System Requirements

- Python 3.8+
- Libraries:
    - Flask
    - joblib
    - scikit-learn
    - plotly
    - sympy
    - numpy

---

3. Project Structure

- main.py: Main application file containing server, model loading, prediction logic, and plotting functions.
- coord_system_model.pkl: Pre-trained Logistic Regression model.
- dataset.json: Dataset containing sample integration questions and corresponding integral setups.
- templates/index2.html: HTML template for frontend rendering.

---

4. Detailed Explanation

Flask App

Initializes a Flask server to handle HTTP requests, process user input, and render dynamic HTML.

```
app = Flask(__name__)
```

Model Loading

Loads a pre-trained Logistic Regression model used to predict the appropriate coordinate system.

```
coord_model = joblib.load("coord_system_model.pkl")
```

Dataset Loading

Reads the dataset.json file containing integration problems and their solutions.

```
def load_dataset(file_path="dataset.json"):
    with open(file_path, "r") as file:
        return json.load(file)
```

Utility Functions

- **extract_equations(text)**: Extracts equations enclosed in double quotes.
- **plot_explicit_surface(eq_func)**: Plots explicit 3D surfaces.
- **plot_implicit_surface(expr_func)**: Plots implicit 3D surfaces.
- **plot_equations(equations)**: Smartly chooses plotting strategy based on the equation type.

Prediction System

- **predict_integral_setup(user_question, predicted_coord_system, dataset)**: Finds the best matching integral setup for the given question and coordinate system.
- **compute_volume(bounds_dict, coordinate_system)**: Computes volume by setting up and solving triple integrals with appropriate Jacobians based on the predicted coordinate system.

Error Handling

Every major processing block (model prediction, parsing, plotting) is enclosed in try-except blocks to prevent crashes and to provide user-friendly error messages.

Web Routes

```
@app.route("/", methods=["GET", "POST"])
def index():
    # Processes user input and returns the rendered page.
```

Handles both GET (initial page load) and POST (form submission) requests.

---

5. How It Works (Workflow)

1. User inputs a query describing a volume computation problem.
2. Model predicts the best coordinate system (Cartesian, Cylindrical, Spherical).
3. Extracts equations from user query.
4. Matches the closest integral setup from the dataset.
5. Computes the volume by symbolic integration.
6. Generates plots of the provided equations.
7. Displays prediction, equation plots, computed volume, and LaTeX output on the webpage.

---

6. Deployment Instructions

1. Install dependencies:

```
pip install flask joblib scikit-learn plotly sympy numpy
```

2. Run the Flask server:

```
python main.py
```

3. Visit http://localhost:81 in your web browser.

---

7. Conclusion

This project bridges machine learning, symbolic mathematics, and interactive visualization into a unified, user-friendly web application. It demonstrates practical applications of natural language processing, mathematical parsing, and Flask development.

The system is designed to be easily extensible for future improvements, such as supporting inequalities, parametric surfaces, and automatic LaTeX rendering.
