Manufacturing Operations API

Overview

This project is a RESTful API designed for predictive analysis of manufacturing operations. It allows users to:

1. Upload a dataset of machine operations.
2. Train a machine learning model to predict machine downtime.
3. Make predictions based on input features like temperature and runtime.

The API is built using **Flask** and **scikit-learn**.

Features

1. **Upload Endpoint (POST /upload)**:
    a. Accepts a CSV file containing manufacturing data.
    b. Saves the file on the server for further processing.
2. **Train Endpoint (POST /train)**:
    a. Trains a Decision Tree Classifier on the uploaded dataset.
    b. Returns model performance metrics, including accuracy and a classification report.
3. **Predict Endpoint (POST /predict)**:
    a. Accepts JSON input with features (Temperature, Run_Time).
    b. Returns predictions for downtime (Yes/No) along with confidence scores.

Technologies Used

- **Python**: Programming language for the API
- **Flask**: Framework for building the RESTful API.
- **scikit-learn**: Library for machine learning model development.
- **pandas**: Data processing and manipulation.

Usage Instructions

1. Upload Dataset (POST /upload)

Request:

- Use a tool like Postman or cURL to send a file.
- Example cURL command:

```
curl -X POST http://127.0.0.1:5000/upload \
-F "file=@/path/to/your/dataset.csv"
```

Response:

```
{
  "message": "File manufacturing_data.csv uploaded success
  "file_path": "data/manufacturing_data.csv"
}
```

2. Train the Model (POST /train)

Request:

- Simply send a POST request to the endpoint.
- Example cURL command:

```
curl -X POST http://127.0.0.1:5000/train
```

Response:

```
{
  "message": "Model trained successfully",
  "accuracy": 0.85,
  "classification_report": {
    "0": {
      "precision": 0.89,
      "recall": 0.85,
      "f1-score": 0.87,
      "support": 20
    },
    "1": {
      "precision": 0.82,
      "recall": 0.88,
      "f1-score": 0.85,
      "support": 15
    },
    "accuracy": 0.85,
    "macro avg": {
      "precision": 0.86,
      "recall": 0.87,
      "f1-score": 0.86,
      "support": 35
    },
    "weighted avg": {
      "precision": 0.86,
      "recall": 0.85,
      "f1-score": 0.85,
      "support": 35
    }
  }
}
```

3. Make Predictions (POST /predict)

Request:

- Send a JSON payload with the input features:

```
{
    "Temperature": 85.5,
    "Run_Time": 120
}
```

- Example cURL command:

```
curl -X POST http://127.0.0.1:5000/predict \
-H "Content-Type: application/json" \
-d '{"Temperature": 85.5, "Run_Time": 120}'
```

Response:

```
{
  "Downtime": "Yes",
  "Confidence": 0.85
}
```

Project Structure

```
project_directory/
├── app.py                # Main application file
├── data/                 # Directory for uploaded datasets
├── models/               # Directory for saving trained mo
├── requirements.txt      # Python dependencies
└── README.md             # Project documentation
```

Future Enhancements

1. **Improve Model Performance**:
    a. Use advanced models like Random Forest or Gradient Boosting.
    b. Address class imbalance in the dataset.
2. **Add Deployment**:
    a. Deploy the API to a cloud platform (e.g., Heroku, AWS).
3. **Extend Features**:
    a. Add more input features like machine age or maintenance history.

Contact

For any questions or issues, please reach out to

+91 7322057950
adityaraj6043@gmail.com