Delivery Prediction API Documentation

This API provides powerful functionality to predict delivery outcomes based on detailed shipment data. By leveraging a pre-trained Gradient Boosting Model, the API ensures accurate and efficient predictions.

Features

- Make predictions on shipment delays using comprehensive shipment details.
- Automatically encodes categorical features with pre-trained label encoders to ensure compatibility.
- Offers intuitive endpoints for both predictions and API health monitoring.

How to Run

1. Clone the repository to your local machine and navigate to the project directory
2. git clone https://github.com/ShubhamPrakash108/On_Time_Or_Delayed
3. cd On_Time_Or_Delayed
4. Install all required dependencies using the following command:
5. pip install -r requirements.txt
6. Start the API server locally with the following command:
7. uvicorn app:app --host 0.0.0.0 --port 8000 --reload
8. After the server starts, open your browser and navigate to the API documentation by pasting the URL below into the address bar:
9. http://127.0.0.1:8000/docs#/default/predict_predict_post

This interactive documentation allows you to test the API endpoints and understand their functionality.

Endpoints

GET /

Description: This endpoint provides a basic health check to confirm the API is up and running smoothly.

Sample Response:

{

"message": "Delivery Prediction API is running. Use /predict endpoint for predictions."

}

POST /predict

Description: This endpoint predicts the likelihood of a shipment delay based on input data provided in JSON format.

Request Body Example:

```
{
  "Origin": "string",
  "Destination": "string",
  "Vehicle_Type": "string",
  "Distance_km": float,
  "Weather_Conditions": "string",
  "Traffic_Conditions": "string",
  "Planned_Delay_days": float,
  "Actual_Delay_days": float,
  "Delay_Ratio": float,
  "Severe_Weather": int,
  "Shipment_Day": int,
  "Shipment_Month": int,
  "Shipment_Year": int,
  "Planned_Delivery_Day": int,
  "Planned_Delivery_Month": int,
  "Planned_Delivery_Year": int,
  "Actual_Delivery_Day": int,
  "Actual_Delivery_Month": int,
  "Actual_Delivery_Year": int
}
```

Response Example:

```
{
  "prediction": float,
  "status": "success"
}
```

Files

1. **app.py**: This file contains the implementation of the FastAPI application, including the GET / and POST /predict endpoints.
2. **gradient_boosting_model.pkl**: The pre-trained Gradient Boosting Model used for making predictions.
3. **label_encoders.pkl**: A collection of pre-trained label encoders for transforming categorical variables into numerical format.
4. **requirements.txt**: This file lists all Python dependencies needed to successfully run the API.

Notes

- Ensure that the required model and label encoder files (gradient_boosting_model.pkl, label_encoders.pkl) are present in the project directory before starting the API server.
- Use the /predict endpoint to submit shipment data and receive accurate predictions in return. Be sure to format the input JSON correctly for smooth processing.
- Running the API locally ensures a quick and seamless experience. Follow the steps outlined above to get started quickly and efficiently.