# Shipment Prediction Project

Chinni Jyothi Prakash

December 21, 2024

# 1 Introduction

This project aims to predict shipment outcomes (whether a shipment will be delayed or delivered on time) using a dataset containing shipment details. The process involved data exploration, preprocessing, model training, evaluation, and deployment of a prediction API. The final solution leverages a Random Forest model due to its superior performance and a user-friendly GUI for interacting with the API.

# 2 Dataset Exploration and Cleaning

## 2.1 Dataset Overview

The dataset contains features such as `Origin`, `Destination`, `Vehicle_Type`, `Distance`, `Weather_Conditions`, `Traffic_Conditions`, and the target variable `Delayed`. Initial inspection revealed 597 null values in the `Vehicle_Type` column.

## 2.2 Handling Null Values

Since `Vehicle_Type` is a categorical variable, the null values were filled with the mode of the column to ensure minimal bias.

## 2.3 Feature Selection

Columns like `Shipment ID`, `Shipment Date`, `Planned Delivery Date`, and `Actual Delivery Date` were excluded as they did not show significant impact on the target variable. Key columns such as `Traffic_Conditions`, `Weather_Conditions`, and `Distance (km)` were retained, as they had clear impacts on shipment delays.

# 3 Exploratory Data Analysis (EDA)

## 3.1 Visualizations

Graphical analyses revealed insights, such as a strong correlation between high/moderate traffic and shipment delays. Visualizations helped identify patterns and relationships between features and the target variable.

### 3.2 Unique Value Identification

Unique values in categorical columns were noted for encoding purposes.

## 4 Preprocessing

### 4.1 Categorical Encoding

Label encoders were used to convert categorical features into numerical representations.

### 4.2 Feature Selection

A final list of features was prepared based on their relevance to the prediction task.

### 4.3 Data Splitting

The dataset was split into training (80%) and testing (20%) sets for model evaluation.

## 5 Model Training and Evaluation

### 5.1 Model Selection

Three models were trained and evaluated:

- Logistic Regression
- Decision Tree
- Random Forest

### 5.2 Performance Metrics

Models were evaluated using metrics such as **Accuracy**, **Precision**, **Recall**, and **F1 Score**.

### 5.3 Results

- Logistic Regression showed relatively lower accuracy.
- Random Forest demonstrated superior performance and was chosen as the final model due to its robustness and accuracy.

| Model | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|
| Logistic Regression | 0.6265 | 0.7029 | 0.8521 | 0.7704 |
| Random Forest | 0.8825 | 0.9414 | 0.8960 | 0.9181 |
| Decision Tree | 0.8620 | 0.9064 | 0.9058 | 0.9061 |

Table 1: Performance Metrics of Logistic Regression, Random Forest, and Decision Tree Models

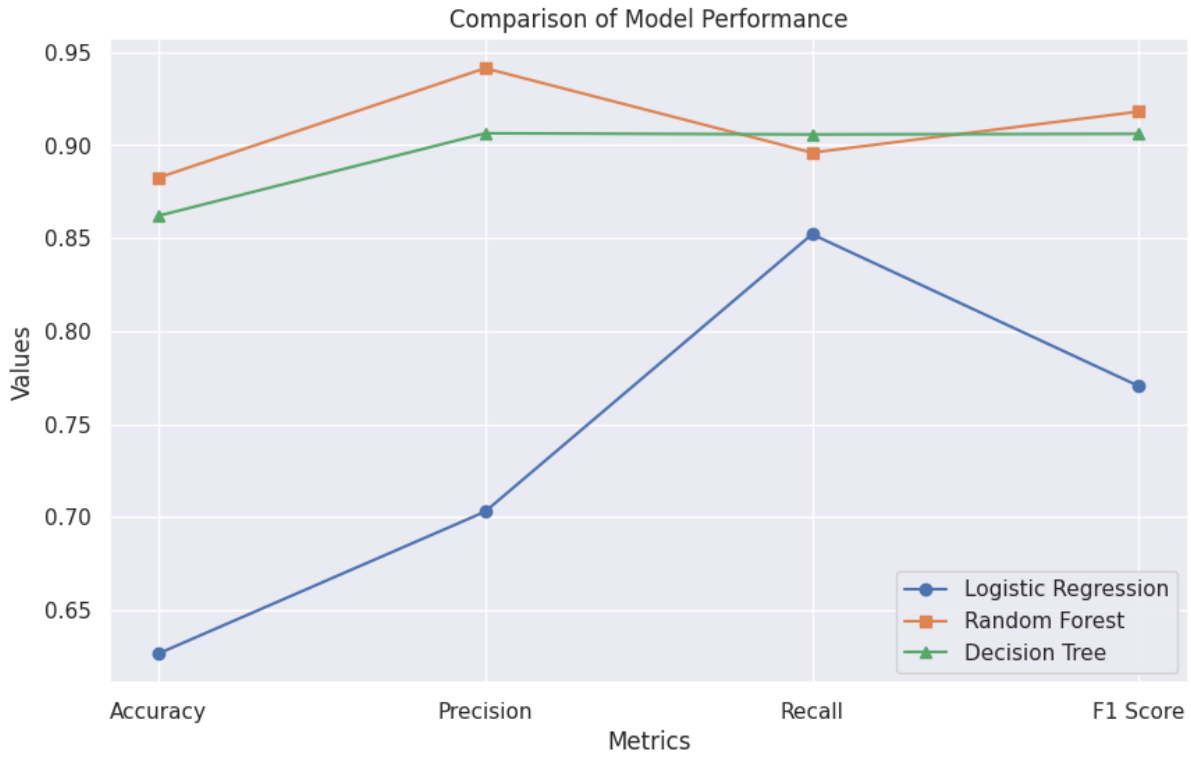


Figure 1: Performance Metrics of Logistic Regression, Random Forest, and Decision Tree Models

### 5.4 Visualization

Performance metrics were visualized to clearly compare the models.

### 5.5 Model Saving

The trained Random Forest model and label encoders were serialized using `joblib` for easy deployment in the API.

## 6 API Development

### 6.1 Setup

An environment was created in Anaconda, and libraries like `FastAPI`, `uvicorn`, and `joblib` were installed.

## 6.2 API Implementation

`shipment_api.py` was created to implement the FastAPI application. Input validation was handled using Pydantic. A `/predict` endpoint was developed to preprocess inputs, use the Random Forest model for predictions, and return results.

## 6.3 Testing the API

A client script (`api_implementation.py`) was created to send POST requests to the API with test data:

```
test_data = {
    "Origin": "Jaipur",
    "Destination": "Mumbai",
    "Vehicle_Type": "Truck",
    "Distance (km)": 1603.0,
    "Weather_Conditions": "Rain",
    "Traffic_Conditions": "Light"
}
```

The predictions were displayed in the terminal.

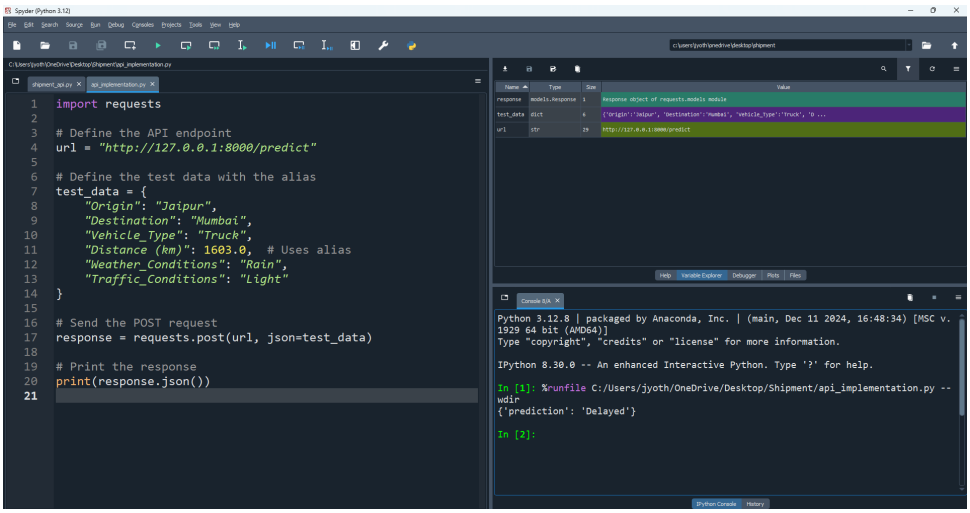


Figure 2: API Test Result

# 7 GUI Integration

## 7.1 Purpose

A small GUI was created using Tkinter to enhance usability for non-technical users.

## 7.2 Functionality

Users can select shipment features like `Origin`, `Destination`, `Vehicle_Type`, `Weather_Conditions`, and `Traffic_Conditions` from dropdown menus. The GUI interacts with the API to provide real-time predictions.

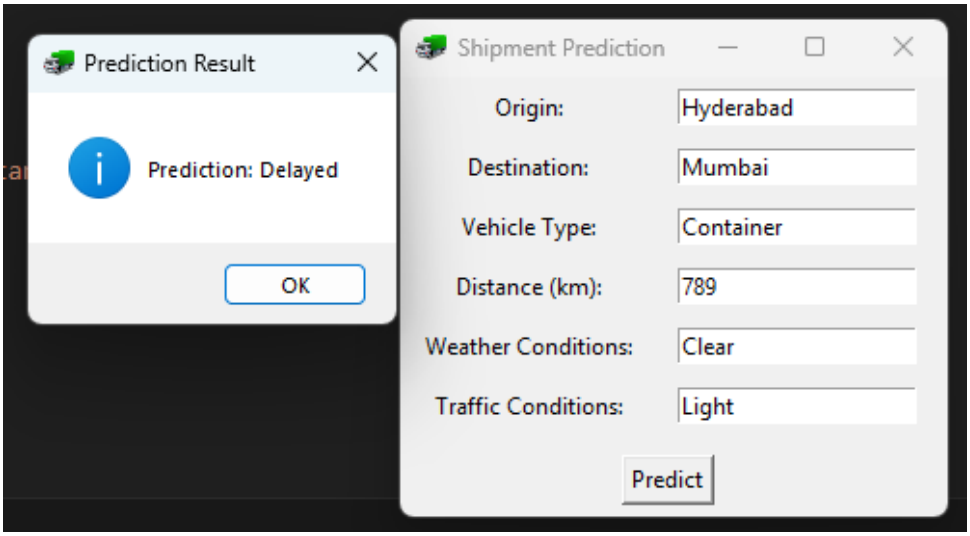


Figure 3: GUI Result

# 8 Key Decisions

- **Model Choice**: Random Forest was chosen due to its ability to handle diverse data types and deliver high accuracy.
- **API Framework**: FastAPI was selected for its speed, ease of use, and ability to create RESTful APIs.
- **GUI Addition**: The GUI was added to improve accessibility for users unfamiliar with APIs.

# 9 Conclusion

This project successfully implemented a shipment prediction system using machine learning. The Random Forest model demonstrated the best performance, and the API and GUI provide a seamless way to interact with the model. Future improvements could include hyperparameter tuning, handling more edge cases, and deploying the API to a cloud platform for broader accessibility.

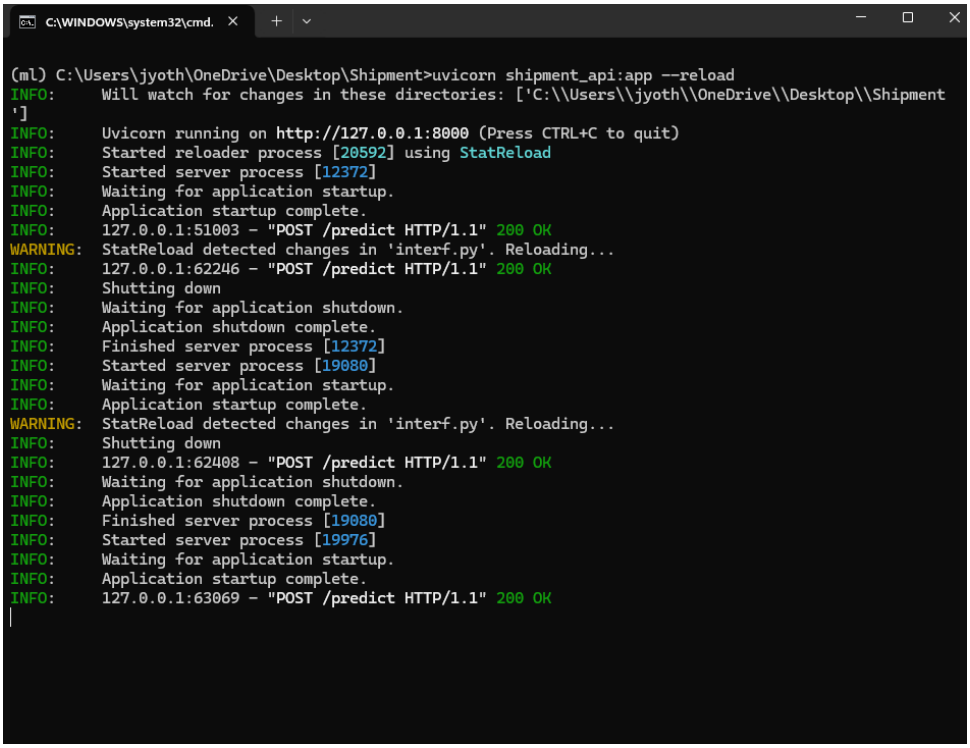


Figure 4: Terminal Result