

AI-Driven Route Optimization and Delay Prediction System for Fleet Operations

Goal:

Predict route delays and deviations in a delivery fleet to improve:

- delivery reliability (better quality)
 - on-time rates (less downtime)
 - optimized route selection (lower energy)
 - driver planning (safer, data-driven work)
-

2. Dataset Overview

Each row = one stop within a route.

We'll predict delays at stop level and then aggregate to route level.

Input Features

RouteID, DriverID, StopID, AddressID, WeekID, Country, DayOfWeek,
IndexP, IndexA, ArrivedTime, EarliestTime, LatestTime,
DistanceP, DistanceA, Depot, Delivery

3. Target Definition

We define two targets for two modelling tasks:

Task	Target Variable	Definition	Type
Classification	delayed_flag	1 if ArrivedTime > LatestTime else 0	Binary classification
Regression	delay_minutes	max(0, ArrivedTime - LatestTime) (in minutes)	Continuous regression

Using classification for "will it be delayed?", and regression for "by how many minutes?".

4. Feature Engineering

Time-based

```
hour_of_arrival → Extract hour from ArrivedTime  
time_window_length → LatestTime - EarliestTime  
delay_ratio → delay_minutes / time_window_length  
weekday_flag → 1 if Monday-Friday else 0
```

Sequence / Route-based

```
stop_deviation → IndexA - IndexP  
distance_deviation → (DistanceA - DistanceP) / DistanceP  
stop_position_norm → StopID / total_stops_in_route  
prev_stop_delay → previous stop's delay (use shift() grouped by RouteID)  
cumulative_delay → cumulative sum of delays within a route
```

Aggregated per Route (for dashboard)

```
route_total_delay = sum of all delay_minutes  
route_delay_rate = mean(delayed_flag)  
route_efficiency = ( $\Sigma$  DistanceP) / ( $\Sigma$  DistanceA)  
avg_stop_deviation = mean(stop_deviation)
```

5.1 Models (Scikit-learn)

These give interpretability and benchmark metrics.

Model	Type	Why Use It
Logistic Regression	Classification	To predict <i>on time vs delayed</i> (binary). Simple, interpretable, baseline accuracy.
Random Forest Regressor	Regression	To predict <i>delay time (in minutes)</i> . Handles non-linear relations like distance, time windows, and traffic day effects.

→ These models help establish KPIs (accuracy, MAE, RMSE) and feature importances before going deep.

5.2 Advanced Model (Using PyTorch)

Model	Type	Why Use It
LSTM (Long Short-Term Memory)	Sequence model	Captures temporal and route order patterns . For example, how earlier stop delays affect later ones in the same route.

Input: Route sequence (`Stop ID`, `Arrived Time`, `DistanceP`, etc.)

Output: Predicted delay (minutes) or probability of delay.

Why this model:

- Routes are naturally *sequential* – each stop depends on the previous one.
 - LSTM can learn *temporal dependencies* and patterns like "if 3rd stop is delayed, 4th likely will be too."
-

Step 2: Technology Stack

Layer	Tool	Purpose
Data Storage	PostgreSQL	Store cleaned route/stop data, model outputs, and KPIs.
Data Processing	Python (Pandas, NumPy, Scikit-learn)	Feature engineering, preprocessing, and baseline models.
Model Training	PyTorch	Build and train the LSTM model for sequence learning.
Backend API	FastAPI	Serve the trained model as a prediction API.
Frontend Dashboard	React.js	Visualize KPIs, delays per route, and comparison charts.
Visualization	Plotly / Chart.js	Create interactive delay maps or performance charts.
Deployment	Docker + GitHub	For reproducible setup and portfolio presentation.

6. Evaluation Metrics (KPIs)

For Classification

Metric	Meaning
--------	---------

Metric	Meaning
Accuracy	% of correct predictions
Precision / Recall	How well we detect delays without false alarms
F1-Score	Balance between precision and recall
AUC-ROC	Measures overall classifier ranking ability

For Regression

Metric	Meaning
MAE (Mean Absolute Error)	Average absolute deviation in minutes
RMSE (Root Mean Square Error)	Penalizes larger delays more
R ²	How much variance in delay the model explains

Business / Operational KPIs

KPI	Description
% of routes with delay > X min	Operational reliability
Avg. delay per km / per stop	Efficiency metric
Route efficiency (DistanceP vs DistanceA)	Indicates deviation impact
Driver delay ranking	Identify training or optimization needs

7. Workflow Pipeline

1. Data Cleaning

Parse timestamps → convert to datetime.

Handle missing `DistanceA` or `ArrivedTime` values.

Ensure numeric columns (e.g., distances) are consistent.

2. Feature Engineering

Derive engineered features from step 4.

Encode categorical features (Country, DriverID, Depot).

3. Split Dataset

Group by `RouteID` → train/test split (e.g., 80 / 20).

Prevent data leakage (same route can't appear in both).

4. Model Training

Train classification model → predict `delayed_flag`.

Train regression model → predict `delay_minutes`.

5. Validation & Cross-Check

Compare baseline vs advanced models.

Perform feature importance analysis.

6. Deployment / Dashboard

Store predictions + metrics in a table (PostgresSQL).

Visualize in dashboard (e.g., Reactjs dashboard).

8. Dashboard

Section	Visualization	Description
Overview	KPI cards (avg delay, on-time %)	Quick snapshot of fleet performance
Route Map	Route deviation map (planned vs actual path)	Visualizes inefficiency
Driver Stats	Leaderboard bar chart	Rank drivers by avg delay per route
Delay Predictor	Input form (day, stops, distance, etc.) → predicted delay	Demo model interactivity
Feature Importance	SHAP / importance bar plot	Explain model transparency

9. Deliverables

Deliverable	Description
GitHub Repo	Full pipeline (data prep, training, evaluation, dashboard)
Reproducible Runs	<code>requirements.txt</code> , <code>main.ipynb</code>
Demo (video)	Short video demo showing dashboard predictions
Report	Problem, Data, Methodology, Results, KPIs, Limitations, Future Work

Deliverable	Description
Active Issues	GitHub issues tracking improvements or bugs
