Final Project Report — Transport Delay Analysis

Abstract

This report summarizes the work performed to analyze and model transport delay using the provided dataset. It covers data cleaning, feature engineering (including weather severity, route frequency, day type), outlier detection and handling (winsorization), modeling (RandomForest and LinearRegression), cross-validation, explainability (SHAP and feature importances), and recommendations.

Table of contents

Cover

Transport Delay Analysis — Final Report

Author: Project Team

Date: December 2025

---

1. Project Goals

- Predict delay minutes and understand drivers of delay.
- Implement robust data cleaning and feature engineering per the project proposal.
- Handle outliers with a transparent policy and document decisions.
- Train interpretable and high-performing models; provide explainability and CV summaries.

2. Dataset

- Source: `dirty_transport_dataset.csv` in repository root.
- Key fields used: timestamps, route_id, passenger_count, weather descriptors, gps coordinates, scheduled times, observed arrival/departure.

3. Data Cleaning

- Timestamp parsing and timezone-normalization.

- Computation of `delay_minutes` (actual - scheduled arrival).
- Semantic invalidation of GPS and passenger counts (e.g., negative passengers set to NaN).
- Imputation strategies for missing values documented in `src/transport_analysis/data_cleaner.py`.

4. Feature Engineering

New features implemented per project proposal:

- `weather_severity`: numeric score derived from weather descriptors.
- `weather_severity_cat`: categorical bucketing of severity.
- `route_frequency`: historical route count in dataset; normalized as `route_frequency_norm`.
- `is_weekend` and `day_type`: weekday/weekend/holiday buckets.
- `scheduled_hour`, `scheduled_minute` parsed from scheduled timestamps.

See `src/transport_analysis/feature_engineer.py` for implementation details.

5. Outlier Detection and Handling

- EDA performed in `notebooks/EDA_Outlier_Analysis.ipynb` — includes IQR and Z-score approaches, semantic invalidation rules, and visualization.
- Winsorization policy: lower quantile = 1st percentile, upper quantile = 99th percentile by default. Applied to `passenger_count` and `delay_minutes` in feature engineering when enabled.
- Winsorization is enabled by default in the export pipeline; refer to `scripts/export_data_view.py` and `scripts/rebuild_outputs.py` (CLI `--no-winsor` to disable).

Rationale: winsorization reduces influence of extreme outliers while preserving data mass and interpretability for downstream models.

6. Modeling

- Models trained: `RandomForestRegressor` and `LinearRegression`.
- Pipeline saves per-model artifacts to `results/` as `model_RandomForest.pkl` and `model_LinearRegression.pkl`.
- Cross-validation (K-fold) applied; mean and std R^2 reported per model and saved to comparison files.

Key artifact: Model performance comparison

7. Explainability and Cross-Validation Visuals

- Multi-model explainability report (SHAP or fallback importances) saved to: model_explainability_report.html.
- If SHAP is available, SHAP summary plots are created; otherwise, model importances or coefficients are shown.
- Cross-validation comparison boxplot included in the report when CV results available: cv_comparison_boxplot.png

8. Results Summary

- Both models produce reasonable predictive performance on the dataset (see `results/model_performance_comparison.csv` and `results/model_performance_comparison.png`).
- Feature importance aligns with domain expectations: scheduled time, route frequency, and weather severity are among top predictors.

9. Limitations

- The dataset is small; results should be validated on larger, more representative data.

- SHAP computation is sample-based to limit runtime; full SHAP analysis may refine insights.
- Winsorization choices (1%/99%) are heuristic; sensitivity analysis is recommended.

10. Recommendations and Next Steps

- Run experiments on a larger dataset and compare winsorization vs. robust estimators.
- Add temporal cross-validation (time-series split) to account for temporal drift.
- Consider causal analysis for policy recommendations.

11. Reproducibility

- To rebuild artifacts:

```
bash python scripts/rebuild_outputs.py
```

- To export engineered CSV with winsorization disabled:

```
bash python scripts/rebuild_outputs.py --no-winsor
```

Appendices

- Notebook: `notebooks/EDA_Outlier_Analysis.ipynb`
- Code: `src/transport_analysis/feature_engineer.py`, `model_builder.py`, `explainer.py`
- Results folder: `results/`

Report generated as a project deliverable draft. Figures and numeric tables referenced above are stored under `results/`.