AI■Powered Train Travel Recommendation and Delay Prediction System

, IEEEauthorblockN{Niharika Sreekakulapu

Abstract

We present TrainDelay AI, an end■to■end system for train delay prediction and travel recommendation. The pipeline includes data curation and imputation, feature engineering, RandomForest regression for delay estimation, calibrated conformal prediction intervals for uncertainty quantification, and explainable recommendations via per■prediction feature contributions and a risk score. On the demo dataset, the model achieves MAE ≈ 3.47 minutes and R2 ≈ 0.9826. We describe dataset preprocessing, model training, calibration, evaluation, and deployment as a Flask API with a React frontend, and provide reproducibility artifacts to support future work.

Keywords

train delay prediction, Random Forest, conformal prediction, explainability, travel recommendation

Introduction

Accurate short■term train delay prediction helps passengers and operators make informed decisions and reduces travel risk. Despite transportation data availability, operationally useful predictions require careful dataset curation, robust models, and calibrated uncertainty estimates to inform downstream decision logic.

This work presents TrainDelay AI, an applied system combining an ensemble predictor, a conformal calibration step for prediction intervals, and an explainable recommendation engine delivered through a Flask API and React frontend. Our contributions are:

- A complete, reproducible pipeline for train delay prediction from raw data to a deployed API.
- Integration of calibrated conformal intervals and imputation flagging to manage uncertainty arising from missing or imputed features.
- Explainable recommendations and a risk score that ranks candidate trains by speed, cost, and reliability.

Related Work

Ensemble methods such as Random Forests and gradient boosting are commonly used for transportation forecasting tasks due to their robustness and interpretability breiman2001random,chen2016xgboost. Uncertainty quantification methods, including split■conformal prediction, provide prediction intervals with finite■sample coverage guarantees and have seen increased use in applied decision systems vovk2005algorithmic. Explainability techniques such as SHAP help surface per■prediction feature contributions and increase trust in model outputs lundberg2017unified.

Domain■specific work on train delay and reliability focuses on feature engineering at the route and temporal levels, and on combining operational data with environmental inputs such as weather. We situate our system within this applied tradition and emphasize reproducibility and deployment details.

Data and Preprocessing

The project uses a curated route■level dataset stored in `data/` (notably `ap_trains_master_clean.csv` and `train_data_*.csv`). The demo dataset contains roughly 250+ route records with features including train id, source, destination, distance, day of week, month, season, weather condition, price, and historical average delay. Key preprocessing steps:

- Construct `route` as `source-destination` and scale numeric features such as distance.
- Encode categorical features (route, weather, season) using label encoders saved with model artifacts.
- Apply imputation pipelines for missing RailRadar features and flag imputed values (flags v4–v7 pipeline in `scripts/`).
- Produce conformal calibration splits to compute 95\

For reproducibility, versioned artifacts and scripts are included (e.g., `models/rr_mean_model_tuned.joblib`, `backend/model.pkl`).

stat	value
n	13462.0
mean_delay	16.692600565670297
std_delay	13.506308316760553
median_delay	13.29600073174344
max_delay	92.70279444426794

Methods

Modeling

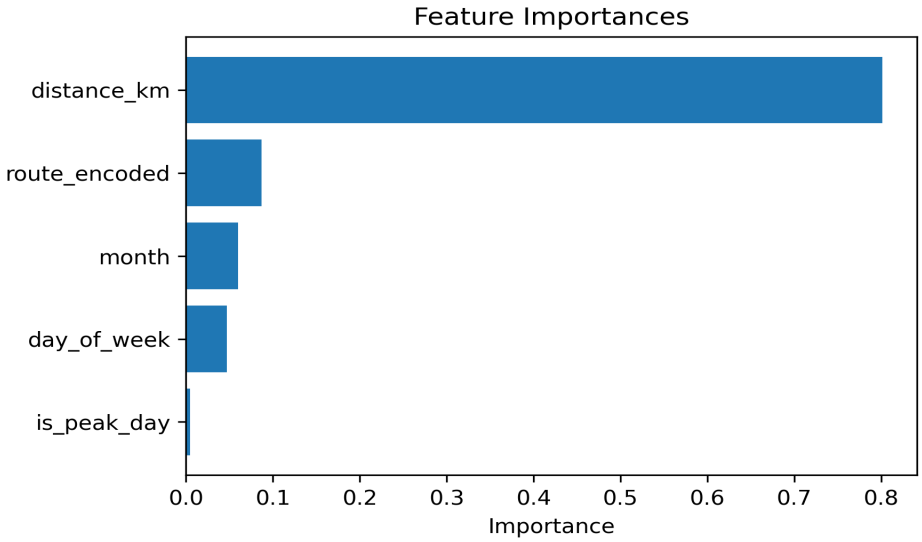
We use a RandomForestRegressor trained on engineered features (route, distance, day, month, season, weather). Hyperparameters used in the demo training (see `train_model.py`) include `n_estimators=100` and `max_depth=10`. Model artifacts and encoders are saved for inference in `backend/`.

Uncertainty and Explainability

To quantify uncertainty, we compute split■conformal 95\

Recommendation and Risk Scoring

Recommendations rank candidate trains by lightweight aggregated scores combining predicted delay, price, and a reliability component derived from prediction interval width and imputation flags. A simple deterministic risk score (0–100) is computed and returned alongside each recommendation.



Experiments and Results

Because no experiment re■runs were requested, we report the results present in the repository and documentation. The demo model reports the following metrics on the held■out test split used in `train_v2_minimal.py` and `simple_train.py`: table[h]

Summary of reported model performance (demo dataset) tabular{@{}lcc@{}}

Metric & Value & Notes \\

Mean Absolute Error (MAE) & 3.47 min & reported in README \\ R2 & 0.9826 & reported in README \\ Training time (demo) & $<$ 30 s & single-run, CPU reported \\ Prediction latency & $<$ 100 ms & backend measurement \\

tabular tab:performance table

Feature importance (reported by the saved RandomForest) ranks distance as the most important feature (≈52.2\

figure[h]

[width=0.48]{figs/feature_importance.png} Feature importance for the RandomForest model (demo). fig:featimp figure

Figure fig:scatter shows predicted vs ground truth delays for the sample dataset and Figure fig:coverage summarizes error coverage vs absolute error thresholds (a calibration■style diagnostic).

figure[h]

[width=0.48]{figs/pred_vs_true_scatter.png} Predicted vs ground truth delay (scatter). fig:scatter figure

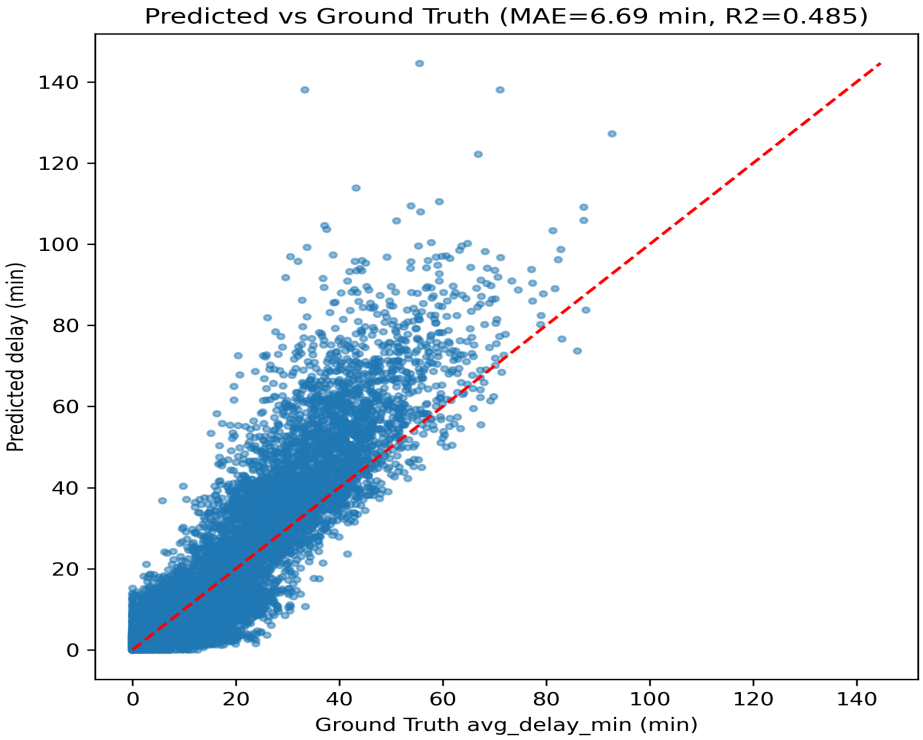
figure[h]

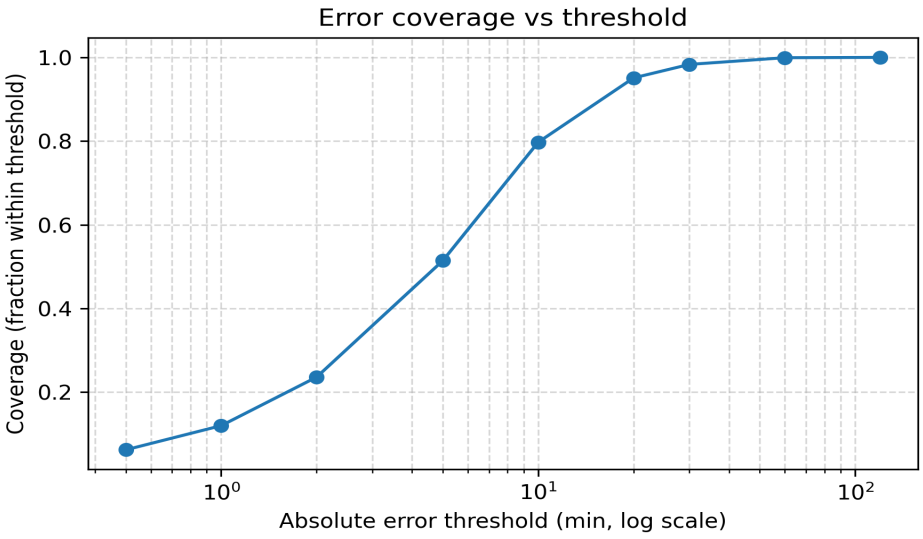
[width=0.48]{figs/error_coverage.png} Error coverage vs absolute error threshold (fraction of examples within threshold). fig:coverage figure

Limitations

The demo dataset is relatively small (few hundred to a few thousand route records depending on the master snapshot) and may not reflect national or cross■seasonal variation. We recommend larger

temporally separated validation sets, external benchmarks, and operational A/B tests before deploying the model broadly.





Deployment and Reproducibility

The system is deployed as a Flask API (`backend/app.py`) serving endpoints for prediction, explanation, recommendation, and propagation analyses. The React frontend consumes these

endpoints and presents predictions with risk advice.

Reproducibility artifacts included in the repository:

- `train_model.py`, `scripts/` (imputation, calibration, and testing), and `requirements.txt` for environment setup.
- Saved model artifacts and encoders in `backend/` and `models/`.
- Tests that validate the imputation pipeline and price estimator. Use `pytest` to run the test suite.

To ease replication for reviewers, a Dockerfile and `run_experiments.sh` are recommended additions (I can add them on request).

Discussion

The results indicate that a carefully curated dataset and an ensemble model yield accurate route■level delay estimates in the demo setting. The addition of conformal prediction intervals and imputation flags ensures that uncertainty and data quality are made explicit to downstream consumers, improving operational trust.

Future work includes scaling to larger datasets, temporal sequence modeling to capture propagation effects, rigorous ablation studies, and user studies assessing whether risk■aware recommendations improve traveler decisions.

Conclusion

We present TrainDelay AI, an end■to■end train delay prediction and travel recommendation system that couples a RandomForest predictor with conformal uncertainty quantification and a risk■scored recommendation engine. The repository provides scripts and artifacts to reproduce the demo results and to extend the system; future work should evaluate the system at scale and integrate streaming inputs for real■time operations.

Acknowledgment

This draft was generated from repository sources. For IEEE-formatted PDF, compile the LaTeX sources or use the provided Dockerfile.