Graph-Enhanced Train Delay Prediction

Nicolas Pinto - Student ID: 807348
github.com/npinto97/dynamic-train-delay-prediction-system

A.Y. 2023-2024

1 Introduction

Train delays are not isolated events but rather the result of a complex interplay of various factors. These factors interact in nonlinear ways, making delay prediction a challenging task. Delays accumulated at a station may propagate to subsequent stations, but not always predictably. In some cases, a train can recover time along its route, whereas in others, delays may amplify due to weather conditions, network congestion, or extraordinary events such as strikes or technical failures.

Furthermore, delays do not only spread along a single railway line but can propagate across the entire railway system. A delay on a highly trafficked route may cause a cascading effect, affecting other lines and compounding scheduling issues.

The **primary objective** of this project is not to develop a predictive model per se, but rather to analyze how integrating additional topological information into a dataset influences the predictive accuracy of an existing model. Instead of relying solely on traditional predictive approaches, the study explores the extent to which structural features, derived from network topology, contribute to improving model performance. By comparing baseline predictions with those enhanced through feature engineering and graph-based methodologies, we aim to assess whether and to what degree these additional inputs refine the model's ability to capture the complexity of train delay propagation. The evaluation also extends to sequential dependencies, leveraging recurrent neural networks (LSTMs) to understand how different feature sets impact time-series forecasting.

The project faces several key **challenges**. The first challenge is the nonlinearity and complex dependencies, as the interactions between different factors influencing train delays are not straightforward. Another challenge is the propagation across the network, since delays can affect interconnected railway lines, requiring models that can incorporate structural dependencies. Time series and sequential patterns present another difficulty, as delays evolve over time, making it crucial to capture temporal dependencies in an effective manner.

2 Dataset and Preprocessing

The dataset used in this project **was constructed from raw data** provided by an online train tracking platform[1], containing detailed information on train schedules, delays, and stops. Data was collected in JSON format for each day and month of the year, providing a comprehensive view of train movements throughout the year 2024.

To facilitate analysis, the dataset was restructured by mapping abbreviated column names to more descriptive ones. Additionally, stop-level information was extracted and normalized into a separate structured dataset, allowing for a more granular analysis of delays at individual stations.

Several preprocessing steps were applied to refine the dataset. **Missing values** posed a significant challenge, particularly for scheduled stops that were later canceled. Instances where a train was expected to stop but did not were removed to maintain dataset integrity. Moreover, certain stops had missing delay values, denoted as "n.d.". These entries were converted to NaN, ensuring compatibility with machine learning models that recognize missing data without introducing bias.

Departure and arrival times at terminal stations presented another preprocessing concern. In the raw dataset, these were recorded as zero, which does not accurately represent **missing time values**. To

[1] https://trainstats.altervista.org/

resolve this, such entries were converted to NaT (Not a Time), the standard representation for missing timestamps in pandas. This adjustment allows models to interpret missing timestamps correctly while preventing misleading assumptions about train schedules.

An **additional feature**, `is_terminal_stop` was introduced to explicitly indicate whether a stop was a terminal station. Validation confirmed that the percentage of missing values in arrival and departure times aligned perfectly with the proportion of terminal stops, ensuring logical consistency in the dataset.

During data inspection, certain entries exhibited highly unrealistic delay values. Some stations recorded extreme negative arrival delays, which were determined to be **erroneous data points**. To prevent such errors from distorting analysis and model predictions, a filtering step was applied to remove outliers based on predefined reasonable delay thresholds.

Furthermore, delay values at terminal stops required special handling. Since trains do not accumulate departure delays at their initial station, and final stations do not require departure delay tracking, missing values in these cases were **filled with zero**.

To enhance predictive performance, redundant and **non-informative features were removed**. Unique identifiers such as train ID were excluded, as they do not contribute to learning patterns. Features that introduce data leakage, such as post-event delay values, were also dropped, ensuring that only available information at prediction time was used. The final dataset retains `stop_arrival_delay` as the target variable.

3 Exploratory Data Analysis

The distribution of arrival and departure delays is highly skewed, with most delays being minor and a few extreme cases. Departure delays tend to be more frequent than arrival delays. Given the interdependence of train networks, delays at one station can propagate, making it crucial to identify congested routes for better predictions. To ensure reliability, only stations with sufficient traffic are considered, **filtering out** those with sparse data that might be affected by outliers.

A strong correlation exists between departure and arrival delays, indicating that initial delays often persist throughout the journey. While some trains recover lost time, others experience worsening delays due to factors like track congestion or weather conditions. Seasonal trends show relatively stable delays in the first half of the year, with increased variability and peaks in July, November, and December, likely influenced by weather, holiday travel, or maintenance. Monthly patterns confirm higher delays in winter and summer, while weekdays exhibit greater variability than weekends. Peak congestion occurs around 6 p.m., aligning with high passenger demand, whereas nighttime delays remain lower due to reduced train frequency.

4 Feature Engineering

To enhance predictive accuracy, we introduced a set of engineered features focusing on time-based characteristics, station-specific metrics, and network connectivity.

We extracted key **temporal features** from the scheduled departure time, including the hour of departure, day of the week, month, and whether the trip occurred on a weekend. Additionally, a rush hour indicator was introduced, marking peak travel periods (7-9 AM, 5-7 PM) to capture congestion effects.

To quantify station congestion, we calculated the frequency of train stops at each station. Stations exceeding the median frequency were labeled as high-traffic hubs, allowing us to distinguish between frequently used and less significant stations.

The railway network was modeled as a graph, where stations represent nodes and train connections form weighted edges. The weight corresponds to the average positive delay, ensuring that only delays influence the network structure. We extracted several key centrality measures:

1. **Degree Centrality**: Captures the number of direct connections a station has, highlighting major transit hubs.

2. **Betweenness Centrality**: Identifies stations that act as critical intermediaries in the network, where delays could propagate.

3. **Closeness Centrality**: Measures how efficiently a station connects to all others, indicating potential delay recovery points.

4. **PageRank**: Weighed by average positive delay, ranking stations based on their influence in propagating delays.

An additional **planned feature** was the integration of weather conditions for each station and each day in 2024. However, due to the extensive data retrieval required, this step was left as a future enhancement to analyze the impact of external factors on train delays.

5 Baseline Model: Linear Regression

Our main goal is evaluate the **differences between the crafted datasets** in predicting train delays at the next stop. This means:

- **Input** (`X`): Features related to train schedules, past delays, station characteristics, congestion, and structural features (if using `df_graph`).
- **Target** (`y`) Next stop arrival delay (`stop_arrival_delay`).

A key research question is whether graph-derived features enhance predictive accuracy. To evaluate this, we compare model performance between a dataset without graph features and one that includes them.

To establish a baseline, we employ **ElasticNet**, a regression model that balances feature selection and regularization via a combination of Lasso (L1) and Ridge (L2) penalties.

A challenge in delay prediction is **data leakage**, where features provide unintended access to future information. Specifically, `stop_departure_delay`, the departure delay recorded at each station, directly correlates with `stop_arrival_delay` and thus cannot be used in real-time forecasting.

To prevent leakage, we shift the departure delay of a train's previous stop to the current row, creating `prev_stop_departure_delay`. This ensures that predictions are based only on past observations.

A key consideration is how to represent `train_number`, a high-cardinality categorical feature (about 16,585 unique values). Direct encoding is impractical due to overfitting risks. Instead, we apply **frequency encoding**, replacing it with the train's average historical delay.

A core issue is the **strong correlation** (0.936) between `prev_stop_departure_delay` and `stop_arrival_delay`. While this feature significantly enhances predictive accuracy, its real-world applicability is limited. In many scenarios, real-time departure delays from prior stops may be unavailable. We adopt a prudent approach: we **exclude** `prev_stop_departure_delay` from our feature set and evaluate the model's performance without it. This forces the model to rely on alternative features and contextual data (`hour`, `day_of_week`, etc.). In other words, by removing `prev_stop_departure_delay`, we explore how alternative features contribute to delay prediction.

5.1 Model Training and Cross-Validation Strategy

ElasticNet's hyperparameters (`alpha` for regularization strength and `l1_ratio` for Lasso-Ridge balance) are optimized using GridSearchCV. However, since train delays exhibit temporal dependencies, standard K-Fold cannot be applied due to potential leakage (training on future data to predict past delays). Instead, we employ **Time-Series Aware Cross-Validation**, where training always precedes testing, ensuring chronological integrity. This mimics real-world forecasting, where past data informs future predictions.

Results As expected, excluding `prev_stop_departure_delay` significantly degrades performance. A comparison of datasets reveals minimal improvement from graph-based features, suggesting that network centrality metrics may not effectively capture delay propagation dynamics.

With optimized parameters (`alpha` = 0.1, `l1_ratio` = 0.9), ElasticNet achieves an R^2 of 0.34, explaining only 34% of variance. While this result is suboptimal, it establishes a benchmark for future enhancements.

6 Advanced Models

6.1 XGBoost

The goal of this phase is to enhance the predictive performance by utilizing the **XGBoost** algorithm. XGBoost is a gradient boosting method that has gained prominence due to its ability to handle complex, non-linear relationships within data.

Also in this case, for model training and evaluation, **TimeSeriesSplit** will be used as the cross-validation technique. To optimize the hyperparameters of the XGBoost model, we will employ **Optuna** [1], a state-of-the-art hyperparameter optimization framework. Unlike traditional grid search, Optuna uses a more dynamic and efficient algorithm to explore the hyperparameter space, making it more resource-efficient and likely to identify optimal settings for improved model performance.

Given the distribution of the traget feature, we decided that **Root Mean Squared Error** is an appropriate evaluation metric. RMSE penalizes larger deviations more heavily than smaller ones, making it a suitable measure for situations where minimizing frequent small errors is important.

The initial experiments with XGBoost show that the graph-enhanced dataset yields only marginal improvements in performance. Specifically, the RMSE, Mean Absolute Error (MAE), and R^2 values improve slightly, but the changes are not significant enough to justify the added complexity of graph-based features.

In a subsequent phase, we sought to integrate **graph embeddings** into the XGBoost model to explore whether the structural relationships learned through embedding techniques such as **Node2Vec** could provide a more effective feature set. To understand how the dataset containing the embeddings was constructed, see the section 6.2

After training the XGBoost model on the base dataset enhanced with Node2Vec embeddings, the results revealed only marginal improvements in predictive performance. Specifically: RMSE remains almost identical to the previous models and MAE slightly improves, the lowest recorded so far. These results suggest that while the graph embeddings capture some relevant latent information about the network, they do not significantly enhance the model's ability to predict delays accurately.

6.2 LSTM

Long Short-Term Memory networks represent a class of recurrent neural networks specifically designed to address sequential data tasks. Their architecture allows them to capture long-term dependencies and temporal patterns while mitigating the vanishing gradient problem that affects traditional RNNs. Given these advantages, LSTMs are well-suited for time-series forecasting problems, making them an appropriate choice for modeling train delays.

This study investigates the predictive power of LSTMs by training models on **three distinct datasets**. The base dataset and the graph-enhanced dataset that we already used above. The third dataset is the embedding-enhanced dataset, constructed using Node2Vec, captures **latent relationships** between railway stations and is further compressed using **Principal Component Analysis** to reduce dimensionality before inputting the data into the model.

A crucial aspect of model training involves proper weight initialization. The applied strategy uses **Xavier uniform initialization** for the weights of linear layers, ensuring balanced variance across layers. A **dropout mechanism** is introduced to mitigate overfitting when multiple layers are present.

Prior to feeding the data into the model, extensive preprocessing ensures that temporal ordering is preserved. The dataset is sorted by key time attributes, and features are **standardized** to have zero mean and unit variance. This step is essential given the sensitivity of LSTMs to scale differences in input features.

Training is conducted using automatic mixed precision, leveraging both feature standardization and gradient scaling to enhance computational efficiency while maintaining numerical stability. The optimization process is carried out using **AdamW**, a variant of the Adam optimizer designed to improve weight decay handling. A **learning rate scheduler** adjusts the learning rate dynamically to promote better convergence. **Gradient clipping** is applied to prevent exploding gradients, a common challenge in training deep LSTM models.

Experimental results indicate that models trained on the graph-enhanced dataset consistently outperform those relying solely on the base dataset. This suggests that incorporating structural information into the model enhances its ability to identify meaningful patterns in train delays. From an

evaluation perspective, the graph-enhanced LSTM exhibits superior performance metrics. The mean absolute error decreases from 2.8948 in the base model to 2.8016, indicating improved predictive accuracy. Root mean squared error is also lower, with values of 6.0194 for the base LSTM and 5.9309 for the graph-enhanced variant, suggesting reduced variance in prediction errors. The coefficient of determination (R^2) rises from 0.4199 to 0.4369, reflecting better explanatory power.

To further investigate the impact of graph embeddings, the model is extended to incorporate Node2Vec-generated embeddings of railway stations. The embeddings are constructed using random walks over the railway network and trained with a Skip-gram model, yielding 64-dimensional representations. These embeddings are subsequently reduced to 10 principal components via PCA, preserving approximately 52% of the variance while maintaining computational efficiency.

The LSTM trained with graph embeddings follows a similar performance trend. The training loss begins at 35.17 and converges to 31.89, while the test loss decreases from 36.86 to 35.41. All the performance metrics fall somewhere between the two models.

These findings suggest that incorporating graph-based features contributes to performance improvements, yet the additional step of integrating full graph embeddings does not yield substantial gains. This implies that while structural relationships enhance predictive power, Node2Vec embeddings alone may not be fully optimized for time-series forecasting.

7 Evaluation and Conclusion

The results obtained from this study should not be interpreted in absolute terms. The overall predictive performance of the models appears suboptimal due to the deliberate exclusion of the `prev_stop_delay` feature, which was highly correlated with the target variable. This decision was made to isolate and analyze the specific contribution of topological features derived from the railway network. Therefore, the primary focus of this evaluation is not on achieving the best predictive accuracy but rather on **comparing the relative performance** of different models trained on various dataset configurations.

To this end, three modeling approaches were explored: ElasticNet, XGBoost, and LSTM, each tested on different dataset variations. The evaluation metrics considered include **Root Mean Squared Error**, which provides an overall measure of prediction error, **Mean Absolute Error**, which reflects the average magnitude of errors, and the **R-squared score**, which indicates how well the model explains variance in the target variable.

The results indicate that models leveraging graph-based features consistently outperform those relying solely on the base dataset. Among the different approaches, XGBoost emerges as the most effective model, with its graph-enhanced variant achieving the lowest RMSE (5.409) and the highest R^2 score (0.487). This suggests that XGBoost efficiently captures non-linear interactions between topological features and train delays. The model trained solely with Node2Vec embeddings, however, does not surpass the one using manually extracted graph features, highlighting that direct structural attributes may be more informative for this task.

LSTM-based models also exhibit improvements when trained with graph features. The Graph LSTM achieves a lower RMSE and a higher R^2 score compared to the Base LSTM. These results suggest that also recurrent neural networks can benefit from explicit structural information, although the extent of improvement remains moderate. Finally, the ElasticNet models, serving as a linear baseline, show minimal gains when incorporating graph-based features, reinforcing the notion that linear models struggle to leverage such information effectively.

The findings of this study underscore the importance of incorporating structural information from the railway network into predictive models for train delays. Graph-based features extracted from the railway network provide a clear advantage over using only standard tabular data, particularly for non-linear models like XGBoost. However, Node2Vec embeddings, while encoding latent network relationships, do not yield significant improvements over explicitly engineered graph features, suggesting that their representation may not be optimally aligned with the predictive task at hand.

These insights open several avenues for future research. First, alternative methods for integrating graph embeddings should be explored, including refining the Node2Vec approach with domain-specific constraints or adopting Graph Neural Networks to learn richer representations of the railway topology.

Another promising direction involves the incorporation of exogenous information, such as weather conditions, which could have a substantial impact on train delays. Initial attempts to collect weather data via API calls were abandoned due to the time constraints associated with large-scale data retrieval.

References

[1] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631, 2019.