Delay Prediction for TGV trains

Machine Learning

AI Mention

Monday 23 October 2023

Team members:
Francisco GARCIA
Samer LAHOUD
Ingrid VALVERDE
Lucas VELOSO
Project Gihub:
Delay Prediction for TGV trains repository

1 Introduction

This report presents the findings and analysis of the “Prediction of TGV Delays” project. The primary goal of this project is to predict the delay times of TGV trains on arrival from january to june 2023 and identify their main causes. In this report, we discuss the preprocessing and analysis of the SNCF dataset, the process of predicting the delay times and their causes and the results obtained.

2 Data Analysis

2.1 Data Collection

The data required for this project was obtained from the SCNF dataset (.csv) of monthly TGV regularity by route. The dataset is available in the french government dataset website.

2.2 Data Features

The dataset includes various features, both categorical and numerical, that provide information about TGV trains.

As a first step, we created lagged features for the initial dataframe based on the departure and arrival stations. This is a feature engineering technique used to capture the temporal dependencies and patterns in time series data.

This step is valuable when predicting train arrival delays and their causes, since lagged features help capture temporal dependencies, assess the relationship between past delays and current ones, model complex dynamics, and provide additional context for predictive models.

2.3 Dataset Split

The training data is used to teach the model, the validation data helps fine-tune and evaluate its performance during training, and the testing data provides an unbiased assessment of how well the model generalizes to new, unseen data. In time series data like train arrival delays, it’s crucial to maintain chronological order in the split.

Therefore, since the original dataset provides monthly information on TGV regularity by route from 2018 to 2023, we randomly separated data from 2018 to 2022 to be included in the training and validation sets. Regarding the testing set, it was created with data from january to june 2023.

The dataset was split into training, validation, and testing sets as follows:

Table 1: Sizes of Training, Validation, and Testing Sets

Set	**Size of X**	**Size of y**
X_{train}	5707	5707
X_{valid}	1721	1721
X_{test}	726	726

This table summarizes the sizes of the training, validation, and test sets for both the input data (X) and the target data (y).

2.4 Exploratory Analysis

We carried out an exploratory analysis to check whether or not there is any missing data, the percentage of this missing data and the breakdown of the number of categories in each categorical variable. The first conclusions are:

- At least 19.2% of the variables in the dataset are categorical. Among the categorical variables, `commentaires_retard_arrivee` displays the highest cardinality with over 200 possible categories. `gare_depart` and `gare_arrivee` have nearly identical cardinalities, both around the 100 mark. In addition to these categorical variables, the dataset also contains continuous numerical and discrete numerical data types, making up 50% and 30.8% respectively.

- The majority of the missing data is concentrated among three categories: `commentaire_retards_depart` with 34.3%, `commentaire_annulation` with 34.3%, and `commentaires_retard_arrivee` with 31.4%. Combined, these three categories account for nearly the entirety of the missing data.

- The data for the variables `commentaire_annulation` and `commentaire_retards_depart` are entirely absent. These variables can be removed from the dataset without compromising its significance.

2.5 Categorical Variables

To further the analysis, we verify the existence of rare categorical variables in the dataset. This is a important step because rare categories can pose challenges for machine learning models. When certain categories have very few occurrences, the model may struggle to learn meaningful patterns, leading to poor predictions for these rare cases.

The distribution of various categories within the categorical variables reveals the following:

- National services account for the majority of travels, representing 85% of the dataset, while international services make up the remaining 15%.

- The distribution of departure and arrival stations exhibits a similar pattern. Notably, Paris Lyon and Paris Montparnasse emerge as the most frequented stations, collectively representing 33% of all journeys. Beyond these stations, no single station commands more than a 5% share. Stations like Paris Est, Lyon Part Dieu, and Marseille St. Charles jointly contribute 12% to the dataset.

2.6 Numerical Variables

We conducted a comprehensive analysis of numerical feature distributions across all instances. This involved visualizing the distributions of each feature, comparing them to normal distributions, and utilizing boxplots to depict data concentration, including median and quartiles, providing insights into central tendencies and spreads.

This analysis helps in identifying outliers and skewed distributions, which are particularly relevant in the context of train delays. Outliers may represent extreme delay cases that require special attention, while skewed distributions could indicate underlying patterns or factors affecting delays.

Understanding the numerical feature distributions aims to equip us with the necessary knowledge to preprocess the data effectively and choose appropriate modeling techniques, enhancing the accuracy and reliability of our predictions for both arrival delays and their associated causes.

- The variables `Distribution_of_retard_moyen_arrivee` and `Distribution_of_retard_moyen_tous_trains_depart` displa a uniform-like distribution primarily consisting of integers.

- Notably, the variables related to departur delays exhibit an exponential distributio pattern, and may be Poisson o Gamma. An example of that for th `Distribution_of_nb_train_depart_retard` is shown in Figure 2.

- In general we see lots of outliers, whic complicates the process of fitting distribution to the data.

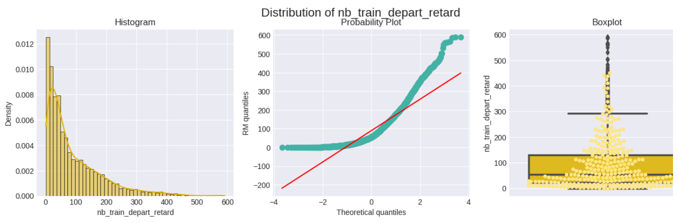

Figure 2: Example of analysis of the distribution of a feature in all instances.

3 Data Preprocessing

In the last section, we delved a little deeper into the dynamics of the problem. In the data preprocessing step, we prepared the variables to facilitate the machine learning process.

3.1 Categorical Variables

3.1.1 Missing Labels

We started by separating the categorical and numerical variables, then proceeded to do the data imputation on these variables. We also added the new "Missing" category, so that the model can capture the reason for the missing instance in case it exists.

Afterwards, we checked the new distribution of some categorical variables that had missing data.

3.1.2 Categorical Encoding

Having added the "Missing" category to the dataset, performing categorical imputation, we then used Decision Tree Encoder (DTE) to try to capture a monotonic relationship between the categories and the target, taking advantage of the encoder's ability to handle high cardinality and complexity in the categorical variables for better model prediction.

We have transformed all the categorical variables into numerical variables by means of DTE, allowing us to have a better metric of how each category acts in relation to the target variable. To do this, we used a heat map to check the correlation between the variables, as can be seen in Figure 3.

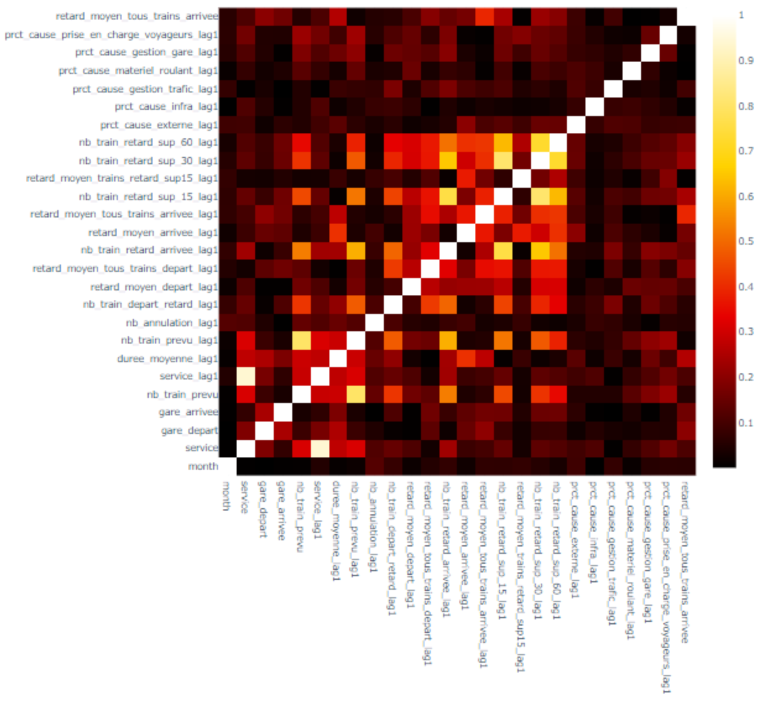

Figure 3: Heat map of the correlation among variables.

We can see that many of the variables are uncorrelated. This is a good thing, as there isn't much redundant information.

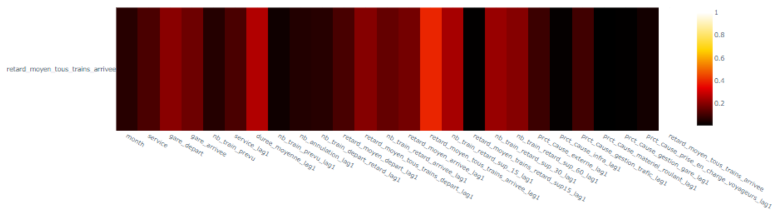

Figure 4: Feature excitation spectrum regarding the mean delay of all trains upon arrival.

After taking a closer look at the correlation between the explanatory variables and the response variable (Figure 4), we notice that some of variables with the highest correlation with the target are: `retard_moyens_tous_trains_arrivee_lag1` (corr=0.39), `duree_moyenne_lag1` (corr=0.26), `nbr_train_retard_sup_15_lag1` (corr=0.24).

3.2 Numerical Variables

3.2.1 Missing Values

The only missing values in the dataset were in the training set due to lag features. These were handled by simply dropping those records using dropna, thereby eliminating the need for imputation. This streamlined the data preprocessing stage, allowing us to concentrate on other facets of feature engineering.

3.2.2 Transformation of variables

Our next objective was to make the variables more symmetrically distributed which helps in training the models. We used the Yeo-Jonhson transformation to improve the asymmetry of the variables. See Figure 5.

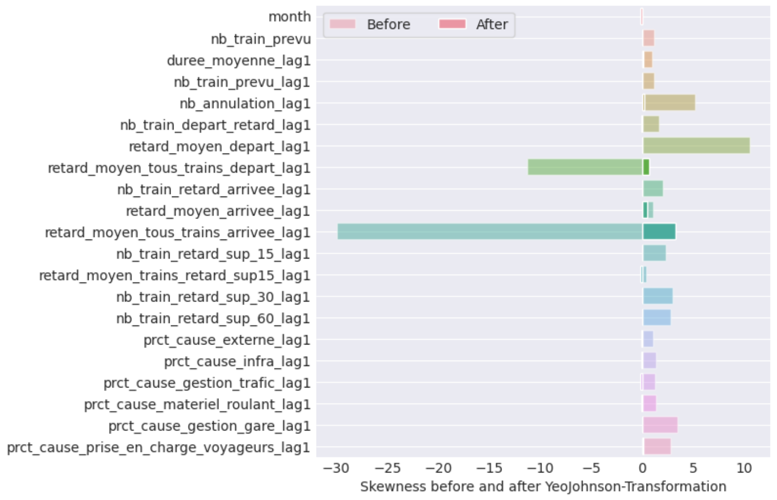

Figure 5: Skewness before and after YeoJohnson-Transformation.

Symmetrically distributed variables are often assumed in statistical and machine learning models, as they can enhance the model's performance by ensuring that it's not unduly influenced by extreme values or skewed data. Addressing asymmetry helps the model better capture underlying patterns in the data, making it more accurate and reliable in predicting both the delay times and their causes.

We then visualized and assessed the distribution of a variable within the dataset. An example of that for the `Distribution_of_nb_train_depart_retard` is shown in Figure 6. We have seen previously in Figure 2 that the distribution was asymmetrical. Now, the distribution although not perfectly normal, tends not to be asymmetrical.

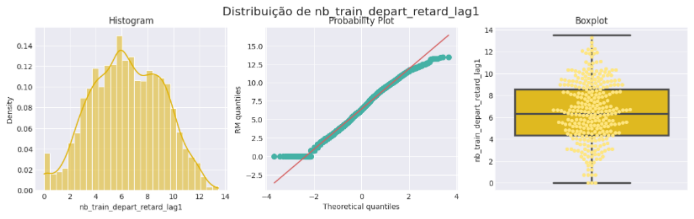

Figure 6: Example of analysis of the distribution of a feature in all the instances.

We also noticed that after the transformations that the variables still have outliers. However, treating these outliers may represent a loss of information. We will therefore choose not to treat these outliers, as removing them could mean a huge loss of information.

3.3 Feature Scaling

As we preferred to keep the outliers, we then used the RobustScaler, which is a feature scaler robust to outliers. This Scaler removes the median and scales the data according to the quantile range.

To verify that the feature scaling process worked, we plotted once again the boxplot of the `Distribution_of_nb_train_depart_retard` variable and verified that most of the samples were between -1 and 1, with the outliers having a module greater than 2.

3.4 Pre-feature selection

3.4.1 Constant and Quasi-Constants Features

The "Pre-feature selection" step, particularly the removal of constant and quasi-constant features (variables with identical values in over 99% of instances), is crucial when predicting train arrival delays and their causes. Such features provide little to no information for predictive modeling and can even introduce noise into the analysis. By eliminating them, we streamline the dataset, reduce computational complexity, and enhance the model's ability to identify relevant patterns associated with delays and their underlying causes.

3.4.2 Duplicated Features

Duplicated features provide redundant information to the model, which can lead to overfitting and computational inefficiency. By eliminating duplicate features during the pre-feature selection phase, we ensure that the model operates with a streamlined and non-redundant set of input variables. This can result in better model performance, improved interpretability, and faster training and inference times.

4 Predicting the Delay Times of TGVs

As an initial step to predicting the delay times of TGVs, we compared twenty models as can be seen in Table 3. We used several performance metrics to assess their effectiveness in the validation set.

Extreme Gradient Boosting (xgboost) demonstrated the best performance with the lowest Mean Absolute Error (MAE) of 1.2583 and a relatively high R-squared (R^2) value of 0.5597, indicating it's a strong predictor of TGV train delay times.

CatBoost Regressor (catboost) also performed well, with a low MAE of 1.1232 and a high R2 of 0.5349, making it another excellent choice for predicting delays.

Huber Regressor, Gradient Boosting Regressor, and Light Gradient Boosting Machine (huber, gbr, lightgbm) displayed similar MAE and R^2 values, suggesting they are competitive options for modeling TGV train delays.

On the other hand, models like Decision Tree Regressor, and some linear regression models (lr, par, en, lasso, llar, omp) showed relatively poor performance with higher MAE and lower R2, making them less suitable for accurate predictions.

We decided to select the Extreme Gradient Boosting (xgboost), the Huber Regressor and the Gradient Boosting Regressor to make an ensemble model. The ensemble model shows very promising performance metrics for the testing dataset (Table 4), with a high R^2 score value of 0.8414, indicating that it explains a significant portion of the variance in predicting train delays. The Mean Absolute Error (MAE) of 0.9016 and the Root Mean Squared Error (RMSE) of 1.4454 are relatively low, suggesting that this model is accurate in its predictions. A low RMSLE of 0.1851 and MAPE of 0.2184 indicate that the model's predictions are precise and have a small margin of error. Moreover, the training time of 3.4100 seconds suggests that the model can efficiently handle the task. Overall, based on these metrics, the ensemble model appears to be a strong performer in predicting train delay times and their causes.

The final evaluation in the testing dataset can be seen in the Figure 7.

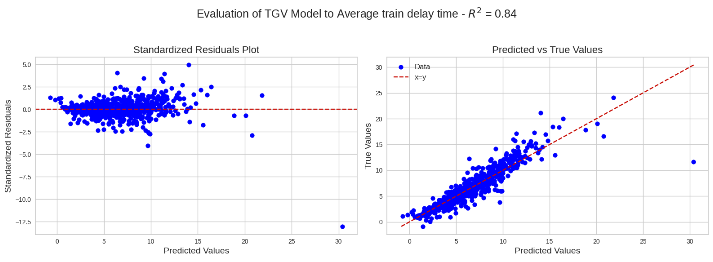

Figure 7: Final evaluation in 2023 data for the Voting Regressor using as base models the , Extreme Gradient Boosting, Huber Regressor, and Gradient Boosting Regressor.

5 Predicting the Causes of the Delays

5.1 Neural Network

In this part, we are trying to predict the reasons that led to the train being delayed each month. As seen before, between six possible categories of delay.

Neural networks, particularly feedforward architectures, are known as universal function approximators. This means they have the capacity to approximate any complex, nonlinear function given a sufficient number of neurons and layers.

Neural networks are capable of automatically learning relevant features from the input data. This feature learning ability is essential in our project, where numerous attributes can influence train delays.

We started by implementing a Feed-Forward neural network using PyTorch. We chose this structure, envisioning the final layer as a softmax function to guarantee that the sum equals one. The forward method applies each layer in sequence to compute the output tensor, making it suitable for a wide range of predictive tasks. This Feed Forward architecture offers flexibility, adaptability, and scalability for various machine learning applications.

Model	Feed Forward
Activation Function	GeLU
Number of Hidden Neurons	512
Number of Hidden Layers	3
Dropout	0.1

Table 2: Feed Forward Model

we initially designed a specialized function to prepare data for our neural network, which efficiently converts the dataset into manageable batches suitable for training. We optimized this process by setting a batch size of 512, streamlining both training and testing workflows. For model evaluation and training, we employed a two-pronged approach. One aspect involved visualizing various performance metrics, such as loss and R-squared, across training epochs. The other facet focused on executing the training regimen, where we adjusted model parameters based on these metrics.

As for computational resources, we configured our system to run on the V100 - GPU. The loss function chosen for optimization was Mean Squared Error, while the optimization algorithm was set to Adam with a learning rate of 1e-6. We performed training over 15 epochs to ensure a thorough evaluation of the model's performance. This strategy was taken to avoid overfitting.

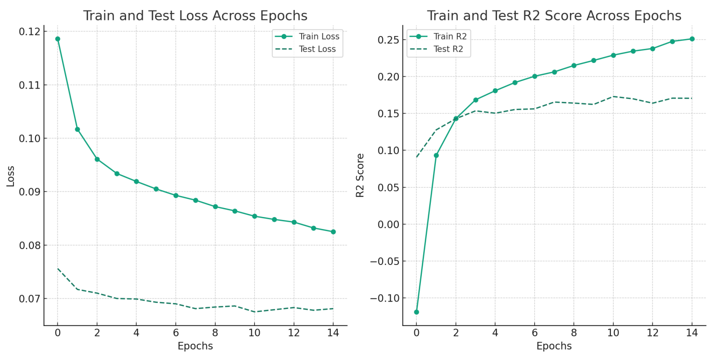

Figure 8: Final Neural Network Evaluation

In graphic 8, the model's performance was evaluated over 15 training epochs using Loss and R^2 score metrics for both training and testing sets. While both the loss and R^2 score showed improvement over the epochs, the relatively low R^2 scores highlight the complexity of the task, which is significantly more challenging than the previous one. The results suggest that enhanced data—such as weather conditions, construction data, and other variables—is needed to improve both generalization and predictive capability.

6 Conclusion

This project aimed to predict TGV train delays and identify their root causes from January to June 2023. We utilized a rich dataset from SNCF, incorporating both categorical and numerical features, and applied robust preprocessing techniques to ready the data for machine learning models. A variety of models were compared for predicting delay times, with Extreme Gradient Boosting (xgboost) showing the most promising results. An ensemble model further improved performance, achieving an R^2 score of 0.8414 on the test set. Additionally, we employed a Feed-Forward neural network to predict the causes of delays. While the model demonstrated learning capability, it also exposed the inherent complexity of the task, indicated by a relatively low R^2 score. This suggests that incorporating more granular data, such as weather and construction information, could significantly enhance the model's predictive performance. Overall, this work provides a solid foundation for future research aimed at optimizing TGV train delay predictions and understanding their underlying causes.

Table 3: Model Performance Metrics for the validation dataset

Model (Abbreviation)	MAE	MSE	RMSE	R^2	RMSLE	MAPE	TT(Sec)
Extreme Gradient Boosting (xgboost)	1.2583	62.3582	5.1756	0.5597	0.2689	0.3600	1.2780
Huber Regressor (huber)	1.4548	60.9491	5.1814	0.5583	0.3121	0.5752	0.8170
Gradient Boosting Regressor (gbr)	1.4502	63.5114	5.2399	0.5531	0.2945	0.5935	2.3030
Light Gradient Boosting Machine (lightgbm)	1.3825	62.5915	5.2492	0.5494	0.3293	0.4979	2.5640
Random Forest Regressor (rf)	1.4407	63.1043	5.2978	0.5373	0.2908	0.5163	5.0760
Extra Trees Regressor (et)	1.2474	63.0847	5.2006	0.5353	0.2703	0.5081	2.4100
CatBoost Regressor (catboost)	1.1232	62.6409	5.1713	0.5349	0.2664	0.3609	6.1840
K Neighbors Regressor (knn)	1.7623	66.5564	5.7126	0.4224	0.3481	0.7290	0.6770
Bayesian Ridge (br)	1.8726	64.0287	5.7699	0.3431	0.4290	0.7940	0.6010
Ridge Regression (ridge)	1.8811	64.0360	5.7726	0.3423	0.4321	0.8060	0.9220
Least Angle Regression (lar)	1.8816	64.0364	5.7728	0.3423	0.4323	0.8068	0.6000
Linear Regression (lr)	1.8816	64.0364	5.7728	0.3423	0.4323	0.8068	0.9720
Passive Aggressive Regressor (par)	2.2280	65.0683	5.9006	0.3084	0.4724	0.9234	1.0750
Elastic Net (en)	2.3421	67.9255	6.1104	0.2694	0.4498	1.1863	0.9870
Lasso Regression (lasso)	2.4509	68.6686	6.2042	0.2335	0.4705	1.2823	0.5900
Lasso Least Angle Regression (llar)	2.4509	68.6686	6.2042	0.2335	0.4705	1.2823	1.0160
Orthogonal Matching Pursuit (omp)	2.7453	68.6470	6.3203	0.1784	0.5217	1.4389	0.8540
AdaBoost Regressor (ada)	3.0483	72.5258	6.4515	0.1656	0.6022	1.9838	0.9840
Decision Tree Regressor (dt)	2.1257	78.7890	6.9829	-0.1743	0.4141	0.6535	1.2030

Table 4: Final Model Performance Metrics for the testing dataset

Model (Abbreviation)	MAE	MSE	RMSE	R^2	RMSLE	MAPE	TT(Sec)
Ensemble Model	**0.9016**	**2.0892**	**1.4454**	**0.8414**	**0.1851**	**0.2184**	**3.4100**