Rapport du Challenge Data : Prédiction du Temps d'Attente des Trains

Delavande Julien
Martin Sébastien

22 mai 2025

1 Introduction

Le but de ce challenge, proposé par Transilien SNCF Voyageurs, est de prédire à court terme le temps d'attente d'un train situé deux gares en amont. L'objectif est de minimiser l'écart entre le temps d'attente théorique et le temps d'attente réalisé à chaque gare sur plusieurs jours. Une attention particulière est portée sur l'évaluation, car les données de test sont disjointes.

Ce rapport détaille les différentes approches testées lors du challenge data. Nous présentons le problème, les techniques d'ingénierie et de sélection des features, les modèles explorés, les stratégies d'optimisation, ainsi que les résultats obtenus.

2 Description du problème

2.1 Structure des Données

Les données sont structurées comme suit :
— **x_train.csv** : 667 265 lignes et 10 colonnes.
— **x_test.csv** : 20 658 lignes et 12 colonnes.
— **y_train.csv** : 667 265 lignes avec la variable cible.

2.2 Origine des Données

Ces données proviennent d'applications métiers collectant les heures théoriques et observées d'arrivées et de départs des trains en gare. Chaque observation représente un arrêt de train spécifique dans une gare donnée un jour précis.

La variable cible **p0q0** correspond à la différence entre le temps d'attente théorique et réalisé. Un score négatif indique un temps d'attente plus long que prévu, tandis qu'un score positif signifie que l'attente est plus courte.

Les variables explicatives comprennent :
— **Variables contextuelles** :
  — train (k) : numéro de train (unique par jour).
  — gare (s) : identifiant de la gare. (3 lettres majuscules - 80 gares uniques)
  — date (d) : date du jour (YYYY-MM-DD).
  — arret : numéro de l'arrêt.
— **Variables passées** :
  — p2q0, p3q0, p4q0 : temps d'attente des trains précédents.
  — p0q2, p0q3, p0q4 : temps d'attente du même train aux gares précédentes.

Les données d'entrainement correspondent à des relévés s'étalant du 3 avril 2023 au 10 novembre de la même année. Les données de test à prédire s'étalent du 11 novembre au 22 décembre 2023.

2.3 Benchmark et Évaluation

La métrique d'évaluation utilisée est la **Mean Absolute Error (MAE)** entre le retard réel et le retard prédit. Le benchmark officiel repose sur un modèle de forêt aléatoire simple et affiche une MAE de 0.89.

Pour nos expérimentations, et en raison du faible nombre de soumissions autorisées par jour (2), nous réalisons une validation croisée sur les données d'entraînement en utilisant une stratégie de validation en 5 plis. Cependant, pour les modèles de deep learning, nous réservons 20% du jeu d'entraînement comme ensemble de validation, tandis que les 80% restants sont utilisés pour l'entraînement.

Étant donné l'aspect temporel des données, nous avons également exploré des stratégies de validation alternatives, telles que la validation par *expanding window* et *sliding window*, afin de mieux refléter la structure séquentielle des données. Toutefois, ces approches ont abouti à des performances similaires aux stratégies classiques de validation croisée, ce qui nous a conduit à privilégier la validation en 5 plis pour la majorité des modèles.

3 Ingénierie et sélection des features et des données

3.1 Les données nominales

3.1.1 Sélection

Dans le dataset, les données **gare** et **train** sont des données nominales. Elles sont respectivement décrites par 3 et 6 lettres majuscules composant les encodages des gares et trains en question. Il existe dans le dataset environ 80 gares et 40000 trains. Seul 2 trains parmi les 40000 apparaissent à la fois dans les données d'entrainement et les données de test du challenge. Pour cette raison, nous avons décidé de ne pas sélectionner les encodages des trains pour la suite du challenge.

3.1.2 Encodage

Plusieurs options s'offrent à nous pour l'encodage de la variable gare :

one-hot encoding Cela consiste à créer une nouvelle variable pour chacune des gares, valant 1 si la gare en question est bien celle indiquée par la nouvelle variable et 0 sinon. Cette approche a été rejetée car elle implique l'ajout de prêt de 80 variables supplémentaires à l'entrée de nos modèles. Un trop grand nombre de variables peut nuire aux performances et à la qualité du modèle.

Label encoding Cela consiste à transformer des variables catégoriques en valeurs numériques. Chaque catégorie unique d'une variable est remplacée par un nombre entier. Bien que cette technique possède le défaut de donner un ordre à nos gares, elle possède l'avantage de rester simple et de n'utiliser qu'une variable d'entrée. Nous utiliserons pour ces raisons cette méthode par la suite, que nous avons implémenté via la méthode homonyme de la bibliothèque sklearn.

3.1.3 Cas particulier du deep learning

Le deep learning offre un avantage particulier : il peut apprendre de lui-même un meilleur encodage que celui offert par un label encoding réalisé durant le prétraitement des données. Pour cette raison, lorsque nous allons utilisé du deep learning, nous modifierons notre architecture et nous ajouterons à notre modèle un module d'encoding pour la variable **gare**, mais également pour la variable **arrêt**.

Ce module sera placé en amont de notre modèle et recevra les gares telles qu'encodées par notre Label encoder classique. Il assignera à chaque label un vecteur appris de plus grande dimension (la plupart du temps de dimension 32) qui nous permettra d'exploiter une structure plus optimale pour cette variable.

3.2 Les dates

Le jeu de données contient également des dates. Initialement, elles ne sont pas exploitables par nos algorithmes. Pour pallier ce problème, nous extrayons des variables que nous jugeons arbitrairement utile à notre problème.

— Une variable allant de 0 à 6 indiquant le jour de la semaine (0 pour lundi, 6 pour dimanche) afin de pouvoir prédire des phénomènes hebdomadaires dans nos données.
— Une variable (redondante à la précédente) qui indique si nous sommes en weekend (1 pour oui, 0 pour non) afin de faciliter l'interprétation des phénomènes liés à l'affluence des fins de semaine.
— Une variable indiquant à quelle semaine de l'année nous sommes ou encore le mois. Cela permet de pouvoir gérer des phénomènes de saisonnalité dans les prédictions.

3.3 Les autres données

Les autres données d'entrée (numéro d'arrêt et temps d'attente à quai pour les trains et gares précédentes) sont utilisées de manière brute car nous les jugeons exploitables dans leur forme initiale.

3.4 Exploration et modification des données

Parmi les données, il a été remarqué que certains retards semblent impossibles à prévoir en raison d'un manque d'information signifiactif dans les entrées. Par exemple : certaines lignes du jeu de données n'ont aucune données particulères (valeur des temps d'attente d'entrée faibles) mais représentent un retard de prêt de 160 minutes. Cette effet peut être expliqué par un blocage du train juste avant son arrivée et nous a incité à essayer à chaque fois de retirer les données imprévisibles du jeu de test lors d'un entrainement.

Nous avons ainsi créer un second jeu de test ou nous avons retiré les valeurs jugées imprévisibles (temps d'attente d'entrée faible mais de sortie élevé).

Il est a noté que ce retard pourrait être expliqué par d'autres variables du jeu et nous offrons donc au algorithme de s'entraîner sur les 2 jeux de données (avec et sans données "imprévisibles") et de garder le meilleur des 2 résultats.

4 Méthodes et Modèles Testés

4.1 Modèles Classiques (Sklearn)

Sans sélection de features, plusieurs modèles ont été testés pour estimer la performance sur les données d'entrainement disponibles, en validation croisée à 5 plis.

Modèle	MAE
Random Forest	0.87
Régression Linéaire	0.98
Gradient Boosting	0.88
LightGBM	0.90
SVR	0.95

Table 1 – Performance des modèles sans sélection de features

Les modèles à base d'arbres, comme Random Forest et Gradient Boosting, offrent les meilleures performances. Cette supériorité peut s'expliquer par leur capacité à gérer la variable catégorielle *gare*, encodée avec un label encoding introduisant une relation d'ordre artificielle. De plus, le grand volume de données d'entraînement favorise ces modèles complexes.

Nous avons ensuite réalisé une selection de features en testant tout simplement la MAE en validation croisée pour tous les sous ensembles de features non vides ($2^n - 1$ avec n le nombre de features, soit pour 10 features - la date étant séparée en mois et jour - 1023 possiblitées), pour le modèle de Random Forest. La meilleure combinaison de features étant : *gare*, *arret*, *p2q0* avec un score en validation croisée de 0.76.

Enfin nous réalisons un grid search classique pour optimiser les hyper paramètres du Random Forest qui nous amène à un score de 0.74 en cross validation sur le train set pour les features suivante :

Hyperparamètre	**Valeur**
n_estimators	400
max_depth	40
min_samples_split	2
min_samples_leaf	4
bootstrap	True

TABLE 2 – Hyperparamètres du modèle

Nous soumettons ce modèle aux données de test et nous obtenons un score de 0.79 sur le leaderboard.

Une autre option a été réalisée en passant le modèle de Random Forest de la régression vers la classification. En supprimant les données jugées hautement imprévisibles, comme mentionner ici 3.4, nous parvenons avec ce modèle à une MAE de 0.70 en local et de 0.72 lors de la soumission en ligne. Battant ainsi les modèles précédemment cités.

4.2 Feature Engineering et Agrégation d'Experts

Un important travail de feature engineering a été réalisé afin d'améliorer la performance des modèles. Tout d'abord, certaines variables ont été transformées, comme l'encodage des gares et la conversion des dates en leurs composantes temporelles (extitjour de la semaine, extitmois, extitannée, extitweek-end ou non). Ensuite, des statistiques agrégées ont été calculées : moyennes et écarts-types des variables cibles par gare et par arrêt. Des nouvelles features ont également été introduites :

— Moyennes glissantes par train.
— Cumulatifs par train.
— Moyennes et écarts-types des features globalement et par jour de la semaine.
— Identification des week-ends.

Une approche de stacking a été explorée en combinant les modèles LightGBM, XGBoost et RandomForest, suivie d'une régression Ridge en méta-modèle. Les performances des modèles individuels ont donné des erreurs MAE de 0.74 pour RandomForest, 0.74 pour XGBoost et 0.76 pour LightGBM. L'ensemble obtenu avec le stacking a permis d'atteindre un MAE de 0.71 bien que cette solution n'ait pas été soumise en raison des meilleurs résultats obtenus dans l'approche suivante.

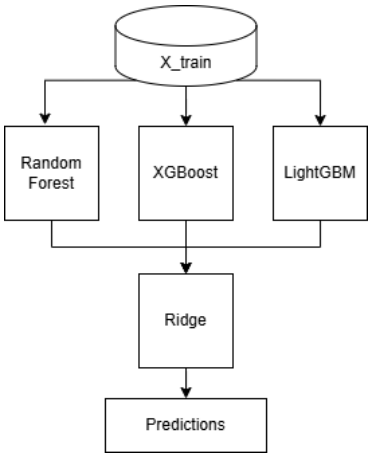

FIGURE 1 – Agrégation d'experts

4.3 Deep Learning

Des architectures neuronales ont été explorées en utilisant les features brutes (sans transformations temporelles), avec un embedding pour les gares et les arrêts (dimension 32). L'utilisation d'un embedding pour l'arrêt, malgré son ordonnancement, permet au modèle d'apprendre une représentation plus flexible et informative, sans forcer un lien linéaire entre les indices des arrêts.

Deux architectures principales ont été testées :

— **MLP simple** : L'embedding de la gare et de l'arrêt est concaténé aux features pxqx, puis passé à un MLP à 4 couches (128, 64, 32, 16, 1) avec activation ReLU, entraîné avec ADAM (lr=1e-3) sur 50 epochs. Cette approche a obtenu un score de 0.62 sur le set de validation et 0.67 sur le leaderboard.

— **Transformers** : Deux couches Transformer ont été appliquées aux features pxqx avant concaténation avec l'embedding, suivies d'un MLP similaire. L'intuition derrière cette approche est de capturer des relations complexes entre les features pxqx à travers les mécanismes d'attention, permettant une meilleure modélisation des interactions. Cette architecture a obtenu un score de 0.68 sur le set de validation.

Enfin, une approche utilisant des CNN pour remplacer les Transformers a été testée, ainsi qu'une alternative purement MLP, sans amélioration notable par rapport aux Transformers. Cette dernière approche a confirmé l'avantage des Transformers dans la capture des dépendances complexes entre les features pxqx.

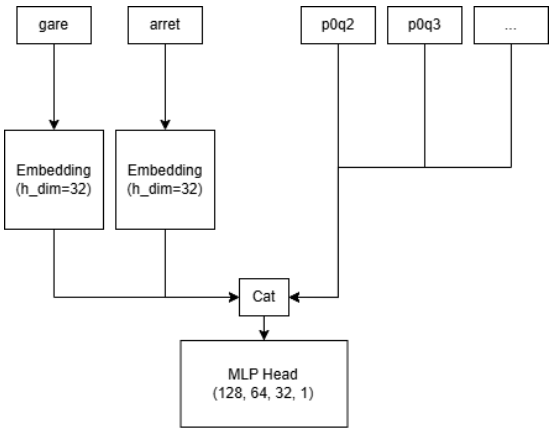

FIGURE 2 – Modèle MLP

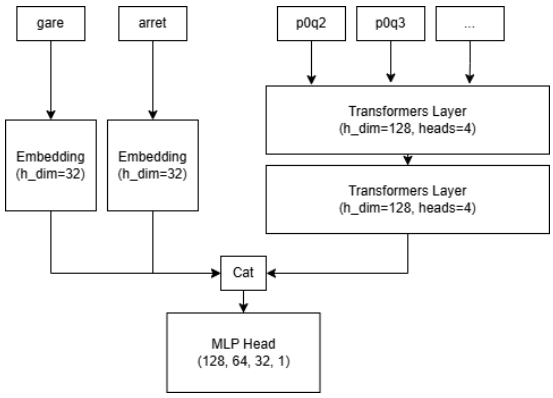

FIGURE 3 – Modèle transformers

En conclusion, bien que les Transformers aient permis une meilleure modélisation des interactions entre les features pxqx, leur gain de performance reste marginal par rapport à la complexité introduite. Le MLP simple s'avère donc être l'approche la plus efficace, offrant un bon compromis entre performance et simplicité, suggérant que la richesse des features pxqx est déjà bien exploitée sans nécessiter de mécanismes d'attention avancés.

5 Résultats et Analyse

— Meilleur résultat atteint sur le challenge (meilleure équipe) : 0.63.
— Soumission à 0.89 : Random Forest sans sélection de features.
— Baseline naïve (prédire 0 partout) : 0.89 (sur le train set).
— Meilleur soumission pour notre équipe (MLP) : 0.77.

6 Conclusion et Perspectives

Ce challenge nous a permis d'explorer plusieurs approches pour la prédiction du temps d'attente des trains, combinant modèles classiques (Random Forest, Gradient Boosting) et deep learning (MLP, Transformers). L'ingénierie des features et la sélection des variables ont joué un rôle clé dans l'amélioration des performances.

Notre meilleure soumission a atteint **0.67 MAE sur le leaderboard**, tandis que la meilleure équipe a obtenu **0.63**. L'approche la plus efficace a été un MLP optimisé, bien que des écarts entre le train et le test suggèrent un sur-apprentissage.

Axes d'amélioration :

— **Amélioration du MLP** : Ajouter Batch Normalization, Dropout et une meilleure régularisation pour limiter l'overfitting.
— **Approche classification** : Convertir le problème en classification multi-classes (discrétisation des retards) pour mieux capturer les distributions pour l'approche MLP.
— **Feature engineering avancé** : Intégrer des indicateurs temporels plus fins dans les approches deep learning.
— **Modèles séquentiels** : Explorer des architectures LSTM ou Transformers spécialisés pour mieux capter la dynamique des retards.
— **Stacking et ensembles pour MLP** : Combiner plusieurs modèles via du stacking ou un modèle méta pour gagner en robustesse en intégrans les modèles de MLP cette fois.

Ce challenge a renforcé notre expertise en machine learning appliqué aux séries temporelles et mis en évidence l'importance du prétraitement des données et du choix des modèles.