Can You Catch It ?

IDASM104 : Projet interdisciplinaire

UNIVERSITÉ DE NAMUR

ALBRECQ Jean, PETIT Antoine,
LAMBART Cyprien, DEGUELDRE Jessica

Professeurs : S. Faulkner B. Frénay V. Salnikov.

2020-2021

Table des matières

Chapitre 1

Introduction

Le but de ce rapport est d'expliquer la démarche et la méthodologie qui a guidé l'élaboration des modèles de machine learning et de l'analyse des données fournie par les *opendata stib-mivb*. Ce rapport est constitué des différentes parties : l'analyse et la préparation des données, l'entraînement des modèles de régression et de classification et l'analyse de leurs résultats. Nous nous sommes concentré uniquement sur une ligne de bus mais le système pourrait facilement être étendu au reste du réseau.

L'étape d'analyse et la préparation des données met en lumière les notions de normalisation, la détecteur d'*outliers*, la selection de *features*. La visualisation des données est également une partie importante de l'analyse des données. En suite dans l'étape d'entraînement des modèles passe par un phase de selection des méta-paramètres et d'optimisation des prédictions.

1.1 Présentation du projet

Il nous a été demandé de développer un nouveau service ou une analyse pertinente par rapport au défis de la mobilité. Plusieurs opendata nous était proposées, nous avons décidé de choisir celle de la STIB. Nous avons choisi de mettre en place un service permettant de savoir si prochain bus qui arrivera à un stop que l'on attend aura du retard ou non.

Chapitre 2

Préparation et visualisation des données

2.1 Récolte des données

La première étape est de toute évidence la récolte des données. Notre projet nous demandais d'avoir accès à un historique de retard mais malheureusement cette historique ne fait pas partie des datasets des opendata STIB. Nous avons donc développé un script python [1] nous permettant de constituer cet historique de retard. Pour obtenir le délais, le script compare le temps d'arrivée théorique (qui nous est fournis par les fichiers GTFS [2]) et l'heure d'arrivée prévue (qui nous est fournie par l'api "*waiting time*"). Le délais est enregistré dans un fichier csv. En plus du délais, le script enregistre la température, la vitesse du vent, l'humidité et la visibilité grâce à l'api OpenWeather [3]. Un nouveau fichier csv est généré chaque jour.

Nous avons dans un premier temps récolté les données pour deux stop (les numéros 0089 et 6608G, voir leur emplacement dans l'annexe A.1) du premier novembre au douze novembre. Les fichiers csv sont disponibles sur le *repository* [4] du projet. Dans un second temps, nous avons récolté les données de tout les stops d'une ligne de bus (la ligne 39). La position de tout les stops de la ligne sont visible sur l'annexe A.2 [5]. Les données récoltées durant cette deuxième phase sont disponible sur le *repository* du projet [6].

2.2 Premier coup d'œil à la structure des données

La commande `data.info()` qu'il y a 12798 lignes sans valeur pour la colonne `delay`. Pour chacune des *features* un graphique du nombre d'occurrences par valeur a été créé (voir l'annexe B.1). On peut voir sur ces graphiques que plusieurs *features* ont toujours la même

1. Le code de ce script est disponible à l'adresse suivante : lien github du script
2. General Transit Feed Specification
3. Documentation disponible ici
4. Disponible ici
5. Une carte GoogleMyMaps est également disponible ici
6. Disponible ici

valeur. On peut également voir que des retards on été enregistré pour d'autres lignes que la numéro 39, il faudra donc supprimer ces dernières.

Le boxplot en annexe B.2 montre la répartition des retards du 19 septembre sur la ligne 39 au stop 0089. On remarque que la majorité des valeurs de délais se situe entre zéro et une minute de retard.

2.3 Préparation des données

Les lignes du dataset pour les quelles la colonne `delay` n'avait pas de valeur ont été supprimée. Les lignes dont la valeur de la colonne `line` n'était pas égale à 39 ont également été supprimée. Les features ayant toujours la même valeur ont été supprimées.

2.3.1 Normalisation

Une fois les données recueillies, nous avons vérifié si elle devaient être normalisés ou non. Après une rapide analyse du dataset généré par le script, il nous a semblé évidant qu'une phase de normalisation était indispensable.

2.3.2 Suppression des outliers

Annexe A

Position des stops

FIGURE A.1 – Position des stops numéro 0089 et 6608G en jaune et noir respectivement.

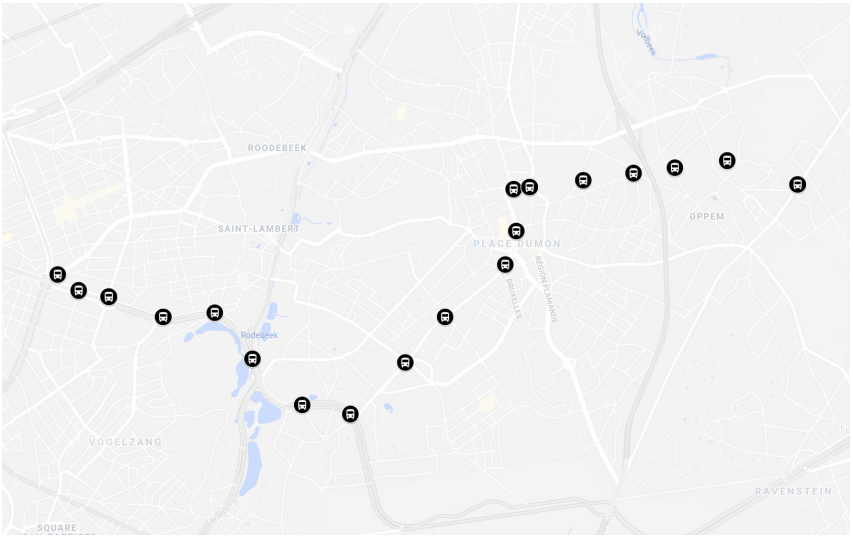

FIGURE A.2 – Position des stops présent sur la ligne 39

Annexe B

Première visualisation des données

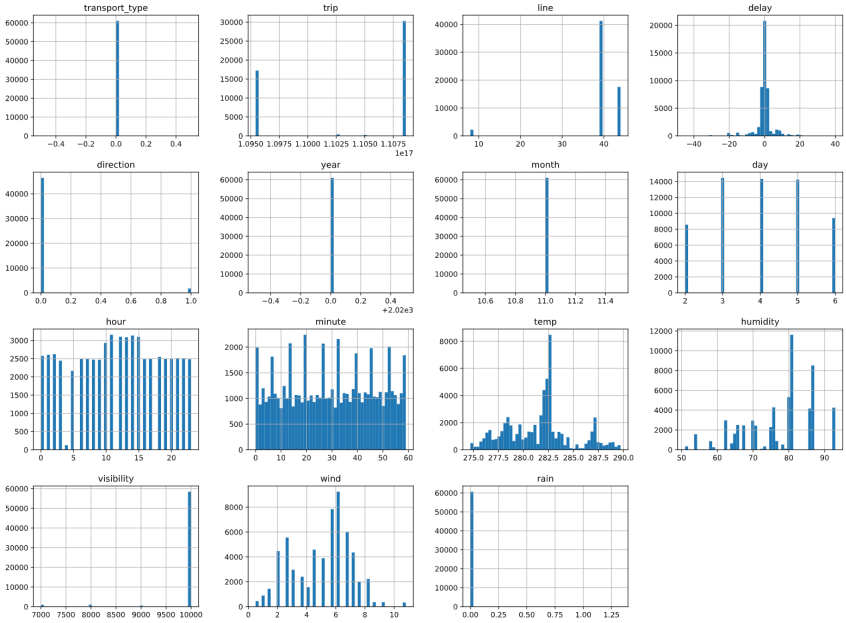

FIGURE B.1 – Plot des features par nombre d'occurrences

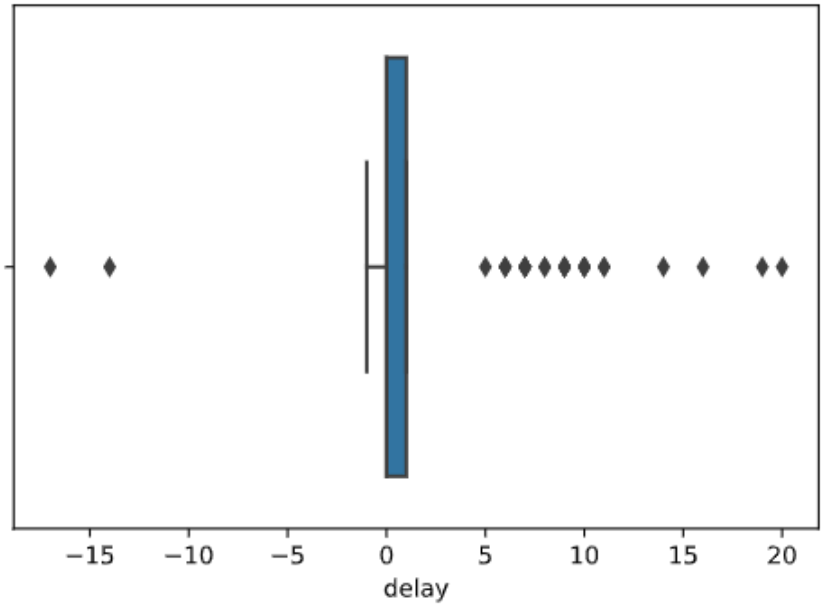

FIGURE B.2 – Boxplot des délais du 19 septembre sur la ligne 39 au stop 0089

Listings

Bibliographie

[1] Robert R. F. DeFilippi. Standardize or Normalize Examples in Python. Avr. 2018. URL : https://medium.com/@rrfd/standardize-or-normalize-examples-in-python-e3f174b65dfc.

[2] Aurélien Géron. Hands-on machine learning. Avr. 2017. URL : http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20%5C&path=ASIN/1491962291.

[3] Urvashi Jaitley. Why Data Normalization is necessary for Machine Learning models. Avr. 2019. URL : https://medium.com/@urvashilluniya/why-data-normalization-is-necessary-for-machine-learning-models-681b65a05029.

[4] Natasha Sharma. Ways to Detect and Remove the Outliers. Mai 2018. URL : https://towardsdatascience.com/ways-to-detect-and-remove-the-outliers-404d16608dba.

[5] Zixuan Zhang. Understand Data Normalization in Machine Learning. Août 2019. URL : https://towardsdatascience.com/understand-data-normalization-in-machine-learning-8ff3062101f0.