ESTIA INSTITUTE OF TECHNOLOGY

BIHAR

HUPI
VALORISER ANALYTIQUEMENT VOTRE DIFFÉRENCE

Rapport de projet tutoré

Supervision de ligne de bus à l'aide du Machine Learning

2022-2023

Encadrants
M. Vincent Moreno
Mme. Kattin Dassance

Etudiants
Dje Bi Mointi Patrice Jean-Marc
Yapi Mpkesso Carole

Sommaire

Introduction 3
A- Compréhension du sujet 3
I- Description du métier 3
1- Description 3
2- Problèmes liés au métier 3
II- Cadre du projet 4
1- Problématique 4
2- Objectif 4
3- KPI d'évaluation 4
III- Données à notre disposition 4
1- Données à disposition 4
2- Données supplémentaires 5
IV- Etat de l'art 5
B- Traitement du sujet 6
I- Pré-traitement 6
II- Analyse exploratoire 8
1- Visualisations 8
2- Etude de la série temporelle 11
3- Etude de la météo 12
4- Corrélation entre variables 13
5- Sélection des features 13
III- Modèles et entraînements 13
1- Préparation des données 13
2- Définition des modèles 14
3- Entraînements 14
4- Evaluation des résultats 15
Conclusion et perspectives 17

Introduction

Le présent rapport traite des résultats et des travaux effectués dans le cadre du projet tutoré soumis par l'entreprise HUPI, et porte sur la prédiction de l'affluence sur des lignes de bus. Ainsi, nous avons commencé par comprendre le sujet qui nous a été soumis et les problématiques autour de la supervision des lignes de bus avant de passer aux études nécessaires pour mettre en place un modèle de prédiction de l'affluence basé sur le Machine Learning.

A- Compréhension du sujet

I- Description du métier

1- Description

La supervision de ligne de bus est assurée en général par une société de transport de bus. Une société de transport de bus fournit des services de transport en bus pour des passagers ou les marchandises. Elle peut être privée ou publique et peuvent offrir des services de transport en commun ou des services transport privé pour des groupes ou les évènements. Dans notre cas, nous nous intéressons au service de transport commun. Les sociétés de transport de bus offrant un service de transport en commun ont pour mission de :

- Assurer la supervision des lignes de bus
- Assurer le transport collectif urbain et interurbain
- Assurer des services réguliers de transport public de personnes
- Assurer des services à la demande de transport public de personnes
- Assurer des services de transport scolaire
- Assurer des services de transport adapté pour les personnes ayant des besoins spéciaux

2- Problèmes liés au métier

La majorité des sociétés de transport en bus rencontre les quatre problèmes ci-dessous :

- Le premier problème est lié au coût des infrastructures. Les sociétés de transport doivent souvent investir dans des infrastructures coûteuses pour améliorer ou étendre leur réseau de transport. Cela peut entraîner des pressions financières importantes et nécessiter des subventions publiques.

- Le deuxième problème est lié à la gestion des passagers et des horaires, car les entreprises doivent souvent faire face à des retards, à des annulations de bus et à des plaintes de passagers insatisfaits. La gestion des horaires peut être particulièrement difficile dans les zones de congestion de la circulation.

- Le troisième problème est lié à la concurrence, car les sociétés de transport doivent souvent rivaliser avec d'autres modes de transport, tels que les trains et les voitures.

- Le quatrième problème est lié à la demande pour les services de transport en commun qui peut varier en fonction des heures de pointe, des événements spéciaux et des périodes de vacances. Les sociétés de transport doivent être en mesure de s'adapter à ces fluctuations pour répondre aux besoins des passagers et il s'agit d'une tâche complexe.

II- Cadre du projet

1- Problématique

Pour une société de transport, être capable de prédire l'affluence des passagers permet de faciliter la résolution du problème lié à la demande du service qui varie en fonction des heures, des évènements etc. Cela permettra par exemple d'optimiser la répartition des bus sur les différentes lignes et de mieux gérer les ressources humaines. D'où la problématique : Comment prédire automatiquement l'affluence des passagers en tenant compte des éléments et des périodes de l'année. Ainsi, le but de notre projet est de proposer un outil basé sur le Machine Learning permettant de prédire l'affluence des passagers.

2- Objectif

Notre objectif vis-à-vis de la problématique sera de mettre en place un modèle de Machine Learning permettant de prédire l'affluence des passagers sur les lignes, par jour, pour les 3 prochains jours.

3- KPI d'évaluation

Comme KPI nous allons utiliser la moyenne de la différence entre les valeurs prédites et les valeurs d'origines de passengerNumber. Cette valeur est fournie par la métrique MAE. Ainsi, on peut évaluer le nombre de moyen de passagers en plus ou en trop pour une ligne de bus ou pour d'autres critères d'agrégation pour évaluer le niveau d'erreur dans la prédiction.

Par exemple, si cette valeur est égale à 0 alors la prédiction est parfaite, cependant si la valeur vaut 2000 alors cela signifie qu'il y a 2000 passagers en plus par rapport aux prédictions ce qui n'est clairement pas une bonne prédiction.

III- Données à notre disposition

1- Données à disposition

Nous disposons d'un Dataset qui les l'historique du cumul journalier de voyageurs allant du 05 avril 2019 au 08 mars 2023. Il est composé de 36901 observations où chaque observation représente l'activité d'un bus pour une journée ; 3 variables explicatives qui sont la date, la ligne de bus et le type de ligne (ligne de jour ou ligne de nuit) et 1a variables prédictives à savoir le nombre de passagers qui représente l'affluence. De plus, il y a au total 39 lignes de bus dont 31 lignes de jour et 8 lignes de nuit.

2- Données supplémentaires

Nous avons effectué des recherches sur les lignes et avons trouvé qu'elles sont réparties par zone :

Zones	Lignes
Lignes zone ouest	05, 16, 18, 25, 45
Lignes zone centre	19, 21, 23, 26, 28, 32
Lignes zone est	08, 09, 13, 14, 29
Lignes interzonales	17, 24, 27, 31, 33, 35, 40, 43, 41
Lignes microbus	36, 37, 38, 42, 39, 46
Services de nuit	B1, B2, B3, B4, B6, B7, B8, B9, B10
Service spéciale	TB6

Ensuite, pour enrichir nos données nous allons ajouter les éléments suivants :

- Les jours férié
- Les jours qui sont weekend
- La météo du jour
- Les jours de vacances scolaires
- Les jours de fête (noël, pâque)
- Les évènements sportifs
- Les évènements culturels
- Les évènements sociaux (Manifestation, élections)
- Les crises (conflit, épidémie)

IV- Etat de l'art

La prédiction des heures d'affluence est importante pour les compagnies de bus car elle leur permet de mieux planifier leurs opérations, de fournir un service de qualité à leurs clients et de maximiser leur rentabilité. Nous avons trouvé 3 approches possibles pour prédire l'affluence des passagers sur les lignes de bus pour les trois prochains jours :

Modèle de régression linéaire : en utilisant l'historique des données de passagers, il est possible de créer un modèle de régression linéaire pour prédire l'affluence des passagers pour les prochains jours. Le modèle pourrait prendre en compte des variables telles que la date, la ligne de bus et le type de ligne de bus, ainsi que d'autres facteurs

tels que les vacances scolaires, les événements locaux et les conditions météorologiques. Nous pouvons utiliser des modèles comme la régression linéaire, la régression logistique et le RandomForestRegressor.

Réseau de neurones : il s'agit du même principe que précédemment mais qui cette ce base sur l'usage des réseaux de neurones. D'après nos recherches les modèles les plus utilisés dans notre cas sont les modèles basés sur les RNN. Nous avons retenu LSTM comme modèle approprié à nos besoins.

Modèles basés sur les séries temporelles : ces modèles exploitent les notions liées aux séries temporelles, dans le sens où elles permettent de tenir compte des tendances, des saisonnalités et des bruits. Comme modèles nous avons retenu : ARIMA (AutoRegressive Integrated Moving Average), SARIMA (Seasonal AutoRegressive Integrated Moving Average), VAR (Seasonal AutoRegressive Integrated Moving Average), Holt-Winters.

B- Traitement du sujet

I- Pré-traitement

Après chargement des données nous effectués les tâches suivantes :

- Vérification de l'existence de valeurs manquantes et doublons
- Vérification et correction des types données
- Extraction des caractéristiques de la date (jour, mois, année)
- Ajout de données supplémentaires évidentes : Météo, Jour férié, période de fête

- **Chargement des données**

```
data_bus = pd.read_csv("data_estia_bihar_passagers.csv",parse_dates=['dateTime']).sort_values("dateTime")
data_bus.shape
```

```
(36901, 4)
```

```
data_bus.dtypes
```

```
dateTime            datetime64[ns]
passengersNumber           float64
line                        object
lineType                    object
dtype: object
```

Lors du chargement des données nous nous sommes assurés du bon typage des dates (avec parse_date) et avons trié le DateFrame selon elle.

- **Vérification de de l'existence de valeurs manquantes et doublons**

Nous avons trouvé qu'il n'y avait pas de valeurs manquantes dans le Dataset mais qu'il y avait 58 doublons.

```
# Vérification de l'existence de valeurs manquantes
n_missval = data_bus.isnull().any(axis=1).sum()
print(f"Il y'a {n_missval} valeurs manquantes")
```

```
Il y'a 0 valeurs manquantes
```

```
# Vérification de l'existence des doublons
n_dbl = data_bus.duplicated(["dateTime","line","lineType"]).sum()
print(f"Il y'a {n_dbl} doublons")
```

```
Il y'a 58 doublons
```

Pour supprimer les doublons nous avons exécuté les codes ci-dessous, ce qui réduit le nombre d'observations de 36901 à 36843.

```
# Suppression des doublons
data_bus2 = data_bus.drop_duplicates(["dateTime","line","lineType"])
n_dbl = data_bus2.duplicated(["dateTime","line","lineType"]).sum()
print(f"Il y'a {n_dbl} doublons")
print(f"Il y'a {data_bus2.shape[0]} observations après suppression des doublons")
```

```
Il y'a 0 doublons
Il y'a 36843 observations après suppression des doublons
```

- **Vérification et correction des types de données**

Après exécution du code ci-dessous, nous avons conclu que les types de données étaient aussi corrects.

```
data_bus2.dtypes
```

```
dateTime            datetime64[ns]
passengersNumber           float64
line                        object
lineType                    object
dtype: object
```

- **Extraction des caractéristiques de la date (jour, mois, année)**

Pour faciliter la manipulation des dates nous avons extrait les caractéristiques de la date, c'est-à-dire le jour, le mois et l'année :

```
data_bus["year"] = data_bus.dateTime.dt.year
data_bus["month"] = data_bus.dateTime.dt.month
data_bus["day"] = data_bus.dateTime.dt.day
```

```
data_bus
```

	dateTime	passengersNumber	line	lineType	meteo_status	year	month	day
0	2019-05-01	4172.0	5	daily	météo correcte	2019	5	1
1	2019-04-06	7330.0	5	daily	météo correcte	2019	4	6
2	2019-04-05	10426.0	5	daily	météo défavorable	2019	4	5

II- Analyse exploratoire

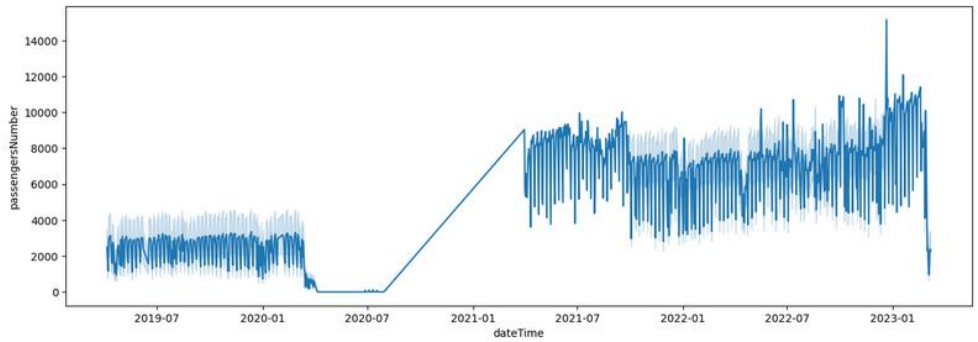
1- Visualisations

- **Visualisation de l'affluence totale par année**

Le lineplot ci-dessous montre l'évolution de l'affluence en fonction de la date.

```
plt.figure(figsize=(15,5))
sns.lineplot(data_bus,x="dateTime",y="passengersNumber")
```

```
<Axes: xlabel='dateTime', ylabel='passengersNumber'>
```

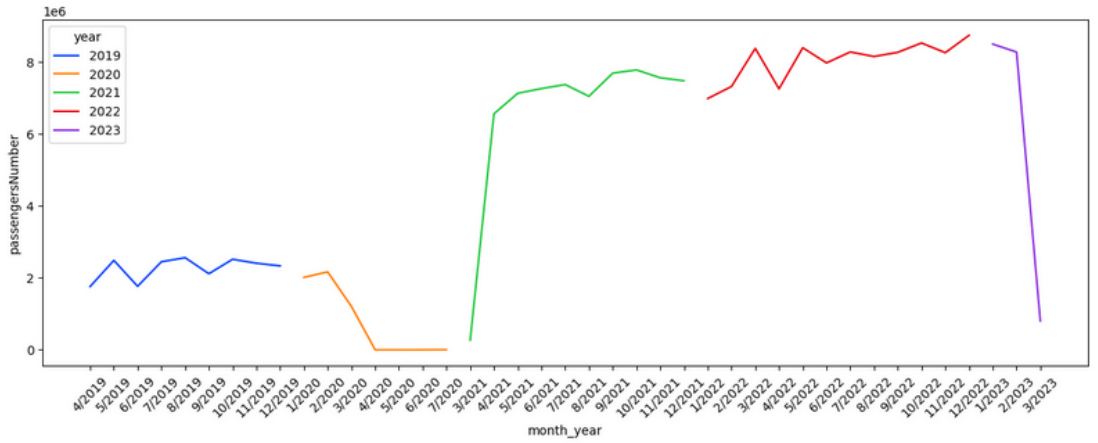


On remarque de 2020 à 2021 il y a un vide. Cela est certainement dû au COVID19. Nous comptons donc supprimer cette partie dans la suite de notre étude. Ensuite on remarque que l'affluence se présente comme une série temporelle.

- **Visualisation de l'affluence par mois de chaque année**

Le lineplot ci-dessus montre l'évolution de l'affluence par mois pour chaque années (de 2019 à 2023) :

```
plt.figure(figsize=(15,5))
sns.lineplot(data=tmp2,x="month_year",y="passengersNumber",hue="year",palette="bright")
plt.xticks(rotation=45)
plt.show()
```



Dans un premier temps on remarque que l'évolution de l'affluence d'une année à une autre n'est pas la même. Mais lorsqu'on observe dans les détails nous remarquons qu'hormis les données incomplètes que l'affluence suit une évolution en saison qui

n'est pas parfaite. En effet, ces variations sont spécifiques à chaque année et ont certainement chacune une explication logique. Ainsi on peut noter les éléments suivants :

- Le COVID19 a entrainé un arrêt de l'activité des bus de mars 2020 à mars 2021, date à partir de laquelle l'affluence à beaucoup augmentée

- L'affluence baisse toujours entre décembre et janvier, puis remonte aussitôt en février sauf en 2023 où au lieu d'augmenter l'affluence à diminuer

- L'affluence augmente toujours entre avril et mai, et entre mai et juillet elle baisse vers juin et augmente vers juillet. Néanmoins, cela n'est pas valable en 2021 où elle augmente légèrement en continue dans la même période

- L'affluence baisse entre juillet et août et augmente entre août et septembre. Mais en 2019, elle a augmenté en août et diminué en septembre

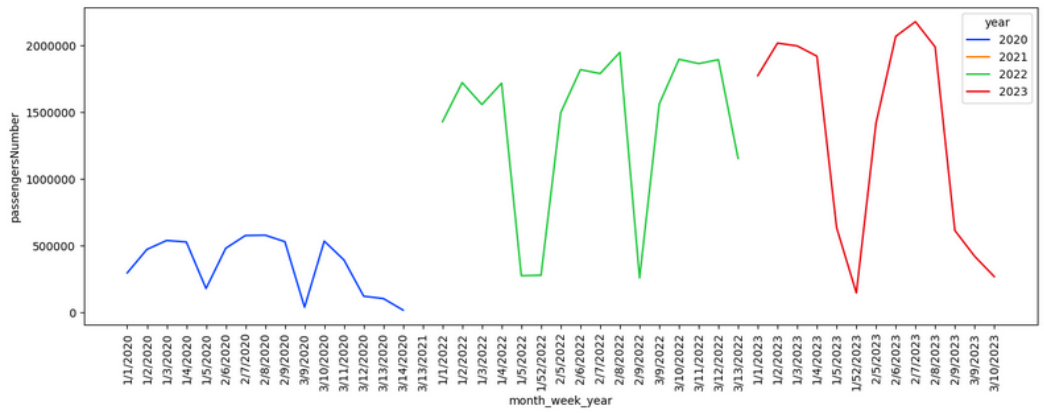
- L'affluence augmente puis baisse entre septembre et novembre, et baisse en septembre. Mais, en 2022 il y a eu une augmentation entre novembre et décembre

Les mois avec les affluences les plus faibles sont janvier et avril tandis que ceux avec les affluences les élevées sont septembre et octobre.

- **Visualisation de l'affluence par semaine de chaque année**

Les lineplots ci-dessous montrent l'évolution de l'affluence par semaine du premier trimestre de chaque. Elle nous permet de comprendre comment l'affluence évolue dans un mois :

```
part1 = tmp3[tmp3.month <= 3]
plt.figure(figsize=(15,5))
fig = sns.lineplot(data=part1,x="month_week_year",y="passengersNumber",hue="year",palette="bright")
plt.ticklabel_format(style='plain', axis='y')
plt.xticks(rotation=90)
plt.show()
```



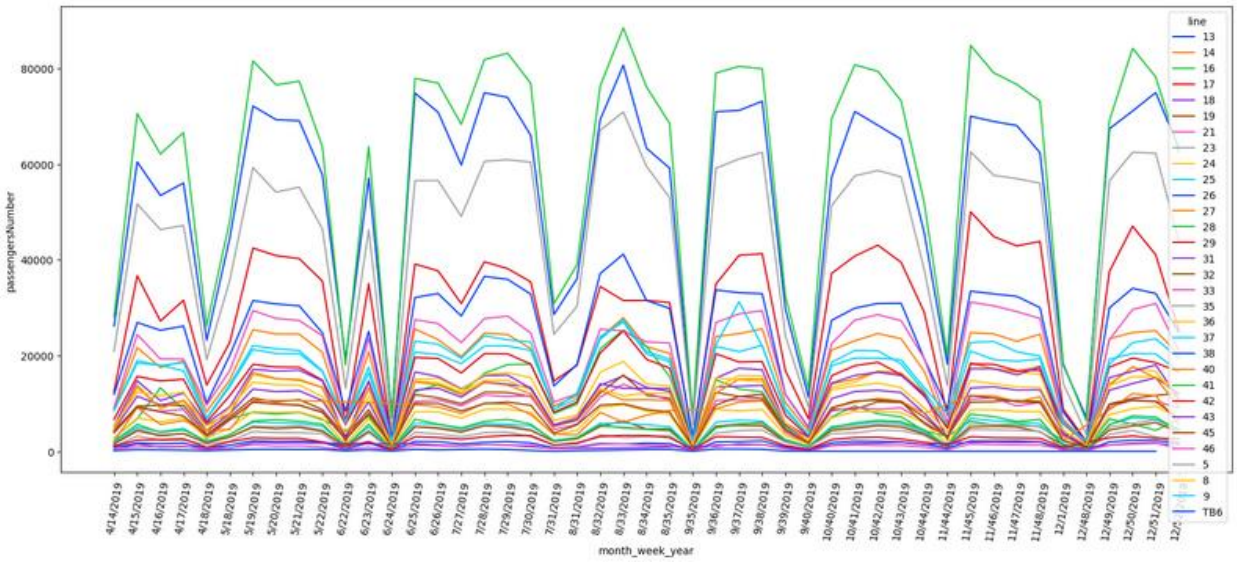
La figure ci-dessous nous permet de remarquer que l'affluence est faible en début de mois puis augmente en fin de mois. Elle varie (augmente ou baisse) selon l'année. C'est donc ainsi que se présente les tendances dans nos données. On peut donc déduire que

toutes les variations inhabituelles sont des bruits qui sont certainement dû à un évènement ou autre phénomène ayant affecté l'affluence.

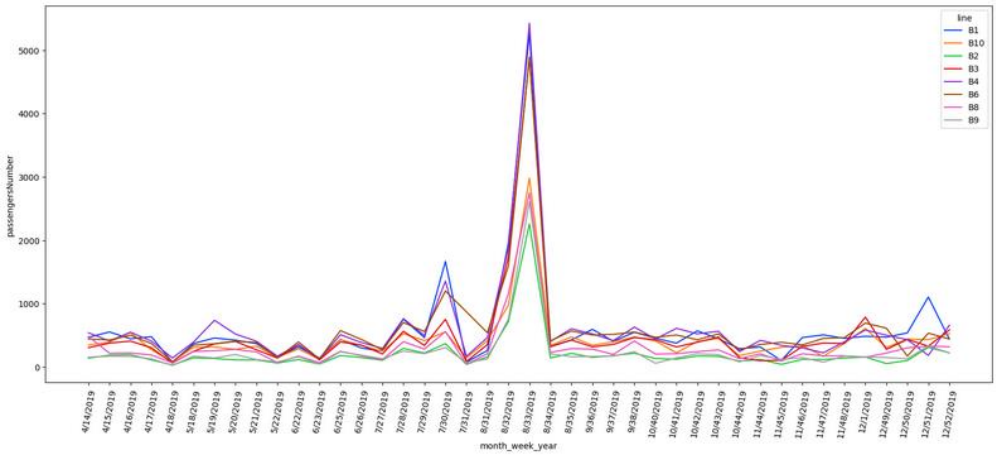
- **Visualisation de l'affluence par ligne de bus**

Dans la suite nous observons différents lineplots qui montre l'évolution de l'affluence par ligne de bus avec agrégation de l'affluence par semaine.

- Affluence pré-covid pour les bus de jour

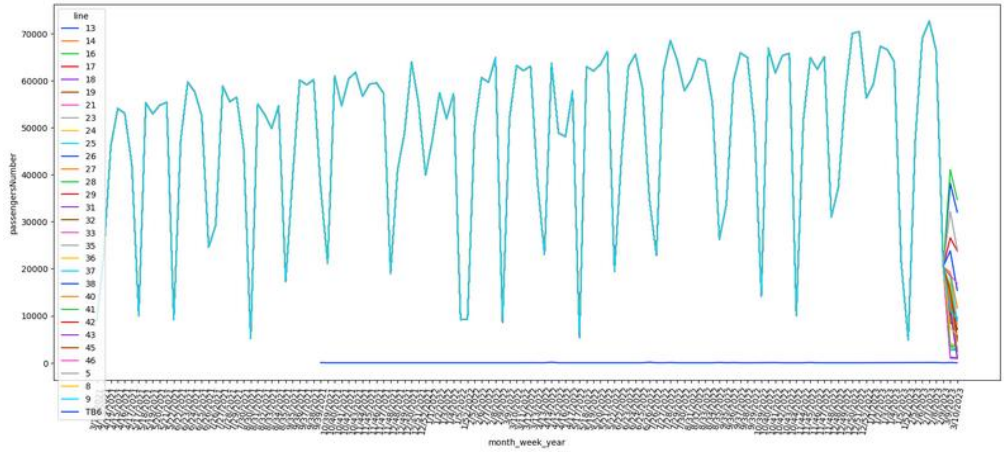


- Affluence pré-covid pour les bus de nuit

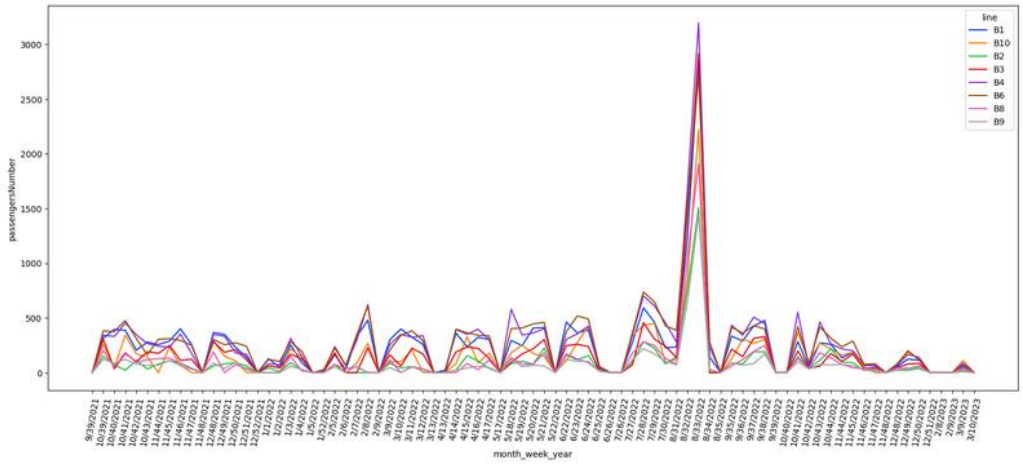


On constate différents niveaux de valeurs de l'affluence pour les différentes lignes, une tendance haute et faible entre chaque début et fin de mois, et différents bruits dans chaque mois. Ainsi on peut déduire que l'affluence est spécifique à chaque ligne de bus et que la tendance normale est une hausse et une baisse de l'affluence entre chaque mois dans lesquelles surviennent des bruits tantôt similaires, tantôt différents. On remarque en particulier qu'il y a un pique dans le mois d'août pour les bus de jour comme de nuit.

- Affluence post-covid pour les bus de jour



- Affluence post-covid pour les bus de nuit



On observe que les lignes de jour possèdent tous la même évolution entre 2021 et mars 2023. Tandis que les lignes de nuit ont une évolution irrégulière avec un grand pique en août 2022.

En conclusion, nous pouvons dire que l'affluence est spécifique à chaque ligne de bus. Les lignes suivent globalement la même progression et parfois ont les bruits (pique haut ou bas) à des périodes précises de l'année. Par ailleurs, les données de la période 2021 à mars 2023 pour les bus de jour semblent incorrectes. On envisage donc de les supprimer.

2- Etude de la série temporelle

Nos données se présentent comme une série temporelle, par conséquent nous allons vérifier s'il s'agit d'une série stationnaire ou non en utilisant le test de Dickey-Fuller augmenté (ADF). Pour ce faire on utilise le fonction adfuller de statsmodels comme suite :

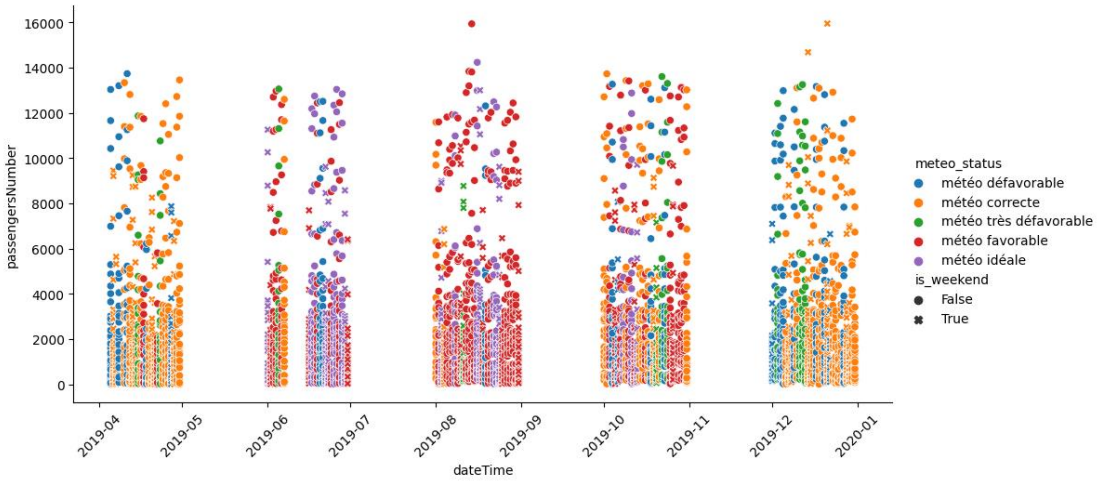
```python
from statsmodels.tsa.stattools import adfuller

def ad_test(dataset):
    dftest = adfuller(dataset,autolag='AIC')
    print("1. ADF : ",dftest[0])
    print("2. P-Value : ",dftest[1])
    print("3. Num Of Lags : ",dftest[2])

ad_test(data_bus["passengersNumber"])
```

```
1. ADF :  -8.12241879574449
2. P-Value :  1.1466000014566834e-12
3. Num Of Lags :  53
```

La valeur de l'ADF indique si la série est stationnaire lorsqu'elle est négativement petite. Dans notre cas, on peut dire que la série est stationnaire. De plus, P-Value indique la probabilité d'observer une statistique de test aussi extrême que celle observée si la série est non stationnaire. Dans notre cas cette valeur est très petite par conséquent on peut affirmer que la série est stationnaire. Par ailleurs, le nombre de décalage (lags) est élevé or plus il est élevé plus la série temporelle possède une certaine autocorrélation, ainsi dans notre cas nos données présentent une autocorrélation. On somme l'affluence est une série temporelle stationnaire qui présente un autocorrélation.

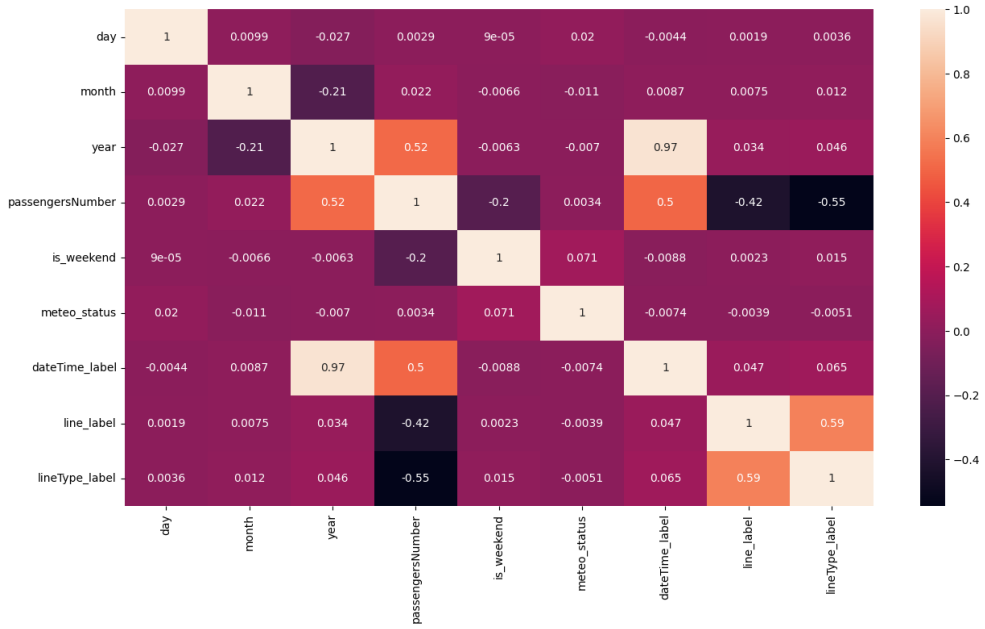
3- Etude de la météo

Le relplot ci-dessous nous permet d'étudier la relation entre l'affluence, la météo et le fait qu'une date soit le weekend ou non.



On remarque qu'il est difficile de dire si la météo influence l'affluence toujours de façon régressive, néanmoins on peut dire qu'il existe un lien entre eux. On remarque aussi que l'opinion sur la météo, même lorsque la météo est défavorable, influence positivement l'affluence par rapport à une météo favorable. Ainsi, on déduit que la météo ne suffit pas seule pour expliquer les piques et baisses de l'affluence.

4- Corrélation entre variables



La matrice de corrélation ci-dessus nous montre qu'il existe un lien intéressant entre, le nombre de passagers et la date. On observe un lien entre la ligne et le type de ligne, ce qui est logique puisqu'il n'existe pas de ligne ayant deux types. Notons aussi que le nombre de passagers est aussi lié au type de ligne plus qu'à la ligne. Il faudra donc bien étudier cette dernière relation. Cependant, toutes les autres corrélations sont assez basses.

5- Sélection des features

On se permet de conserver l'ensemble des variables sans faire de transformation, ni d'ajout de données, on sélectionne donc toutes les variables. Ainsi, les features que nous avons sélectionné sont les suivantes :

- year: L'année
- month : Le mois
- day : Le jour
- line : Ligne
- lineType : Type de ligne

III- Modèles et entraînements

1- Préparation des données

Avant de passer à l'étape des entrainements nous avons commencé par séparer notre jeu de donnés en deux, l'un pour les tests et l'autre pour l'entrainement. Nous avons utilisé les données de la période 2019 et 2022 pour le jeu d'entraînement et celles de 2023 pour les tests :

Training Set	Données de 2019 à 2022
Test Set	Données de 2023

Ci-dessous un exemple de code de la séparation des données :

```
# Division des données
data_test = data_bus[data_bus.year == 2023]
data_train = data_bus[data_bus.year < 2023]
X_train = data_train[features]
y_train = data_train["passengersNumber"]
X_test = data_test[features]
y_test = data_test["passengersNumber"]
print("Données d'entrainement : ",data_train.shape[0])
print("Données de test : ",data_test.shape[0])
preprocessor.fit(X_train)
```

```
Données d'entrainement :  34746
Données de test :  2097
```

Pour le prétraitement des données nous avons utilisé un pipeline qui contient les transformations ci-dessous :

Transformation	**Variables**
OneHotEncoder	["line","lineType"]
passthrough	["year","month","day"]

2- Définition des modèles

Nous présentons ici les modèles que nous allons utiliser pour les entrainements en précisant leur définition.

Modèle	**Type**	**Librairie**
RandomForestRegressor	Modèle de régression	sklearn
ARIMA	Modèle de série temporelle	statsmodels
LSTM	Modèle de réseau de neurones	keras

Pour l'optimisation des hyperparamètres nous avons utilisé des méthodes différentes pour chaque pour chaque modèle :

Modèle	**Méthode d'optimisation des hyperparamètres**
RandomForestRegressor	GridSearch
ARIMA	auto_arima
LSTM	Pas de méthode utilisé

3- Entraînements

Pour entraîner nos modèles nous avons utilisé la méthodologie suivante :

1- Transformation des données
2- Entraînement du modèle avec une méthode d'optimisation, si nécessaire
3- Réentraînement du modèle avec les hyperparamètres optimisé, si nécessaire
4- Prédiction sur les données
5- Stockage des résultats

Ci-dessous les détails d'entraînement pour chaque modèle :

Modèle	Hyperparamètres	Valeurs
RandomForestRegressor	n_estimators	100, 150, 200
ARIMA	(p,q,d)	De 0 à 2 pour chaque paramètres
LSTM	Aucun	Aucun

Concernant LSTM nous avons définit son entraînement comme suit :

```
# Préparation des données
X_train_scaled = preprocessor.transform(X_train).toarray()
X_train_scaled_reshape = X_train_scaled.reshape((X_train_scaled.shape[0], 1, X_train_scaled.shape[1]))

# Entrainement
model = Sequential()
model.add(LSTM(50, activation='relu', input_shape=(1, X_train_scaled.shape[1])))
model.add(Dense(1))
model.compile(optimizer=Adam(learning_rate=0.01), loss='mse')
model.fit(X_train_scaled_reshape, y_train, epochs=50, batch_size=16, verbose=1)
```

4- Evaluation des résultats

Pour l'évaluation des modèles nous avons utilisé les métriques ci-dessous :

- **RMSE** : Pour mesurer l'erreur
- **MAE** : Pour mesurer l'écart moyen entre les valeurs prédites et les valeurs d'origines (Dans notre elle mesure le nombre de passager en trop ou en moins)
- **R^2** : Qui mesure la qualité de l'ajustement des données

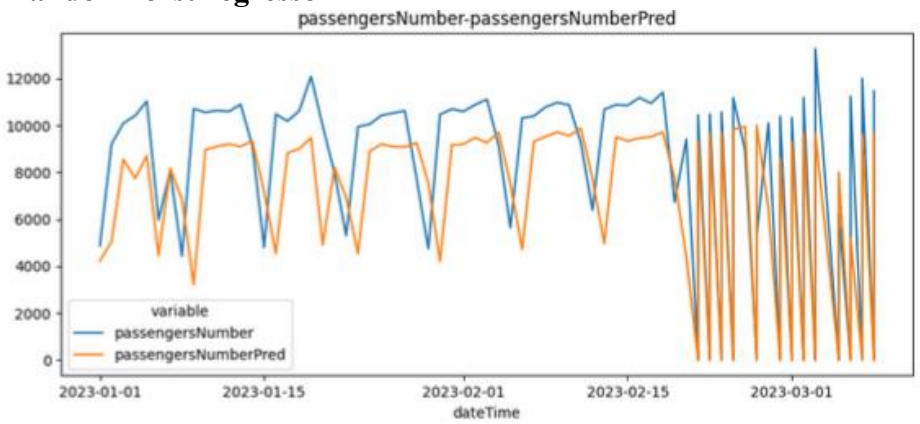
Les résultats ci-dessous sont ceux obtenu sur les données de test :

Modèle	Hyperpramètres	RMSE	MAE	R^2
RandomForestRegressor	n_estimator=100	2371.91	3195.56	0.14
ARIMA	(p,q,d) = (1,1,1)	4589.94	4143.26	-0.78
LSTM		4187.02	3722.34	-0.48

D'après le résultat ci-dessous **RandomForestRegressor** est le meilleur modèle. Néanmoins, nous pouvons dire que globalement les résultats ne sont pas bons. En effet, les marges de MAE sont trop grandes et les ajustements trop bas. Cela doit être certainement dû à une mauvaise configuration des modèles et au manque de niveau des variables explicatives. Une solution serait d'ajouter des variables évènementielles et restreindre l'entraînement à différentes lignes de bus. Ci-dessous, les lineplots qui

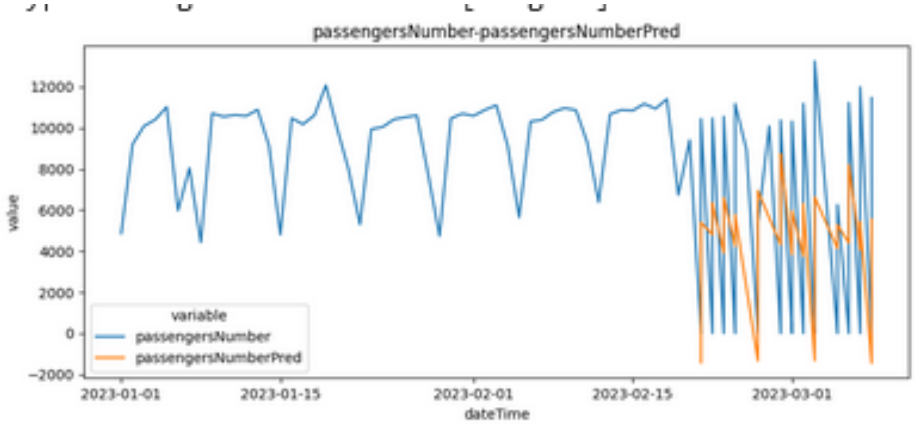
permettent de comparer les prédictions avec les variables cibles.

- **RandomForstRegressor**



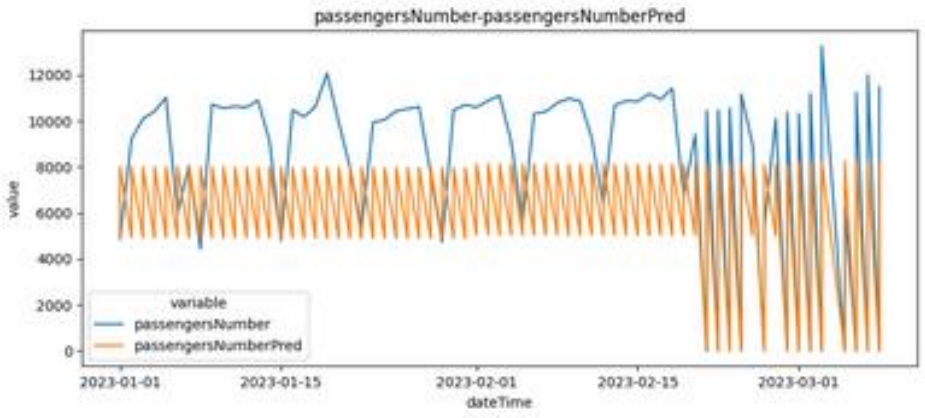
La progression de l'affluence est correcte mais les prédictions sont mal ajustées.

- **ARIMA**



Le modèle ne s'est pas ajusté aux données de l'affluence et ne suit pas sa progression.

- **LSTM**



Le modèle ne s'est pas ajusté aux données mais arrive à suivre leur progression contrairement à ARIMA.

Conclusion et perspectives

L'approche que nous avons présenté dans ce rapport repose sur l'usage d'un modèle général qui n'intègre pas de données évènementielles même si nous en avons étudié certaines. Seul un modèle basé sur RandomForestRegressor nous a permis d'avoir des résultats intéressants. Une solution possible pour améliorer nos résultats aurait été de spécialiser nos modèles sur différente ligne de bus et d'ajouter des données évènementielles. Comme difficulté, nous avons rencontré des difficultés pour effectuer les bonnes visualisations et pour utiliser et optimiser les modèles ARIMA et LSTM. Dans l'ensemble nous avons appris à étudier des séries temporelles et compris les contraintes liées à la résolution de ce genre problème et avons retenu que contrairement à résolution des sujets classiques il est encore plus important tenir de l'influence des évènements extérieurs sur les données.