



PYTHON FOR DA SEOUL BIKE SHARING DEMAND

Thomas DUVAL – IBO 2

https://github.com/thomasduv/Seoul_Bike_Sharing_Demand

INTRODUCTION



Ce projet s'inscrit
dans le cours de
Python for DA



Le dataset d'étude est
Seoul Bike Data
<https://archive.ics.uci.edu/ml/datasets/Seoul+Bike+Sharing+Demand>

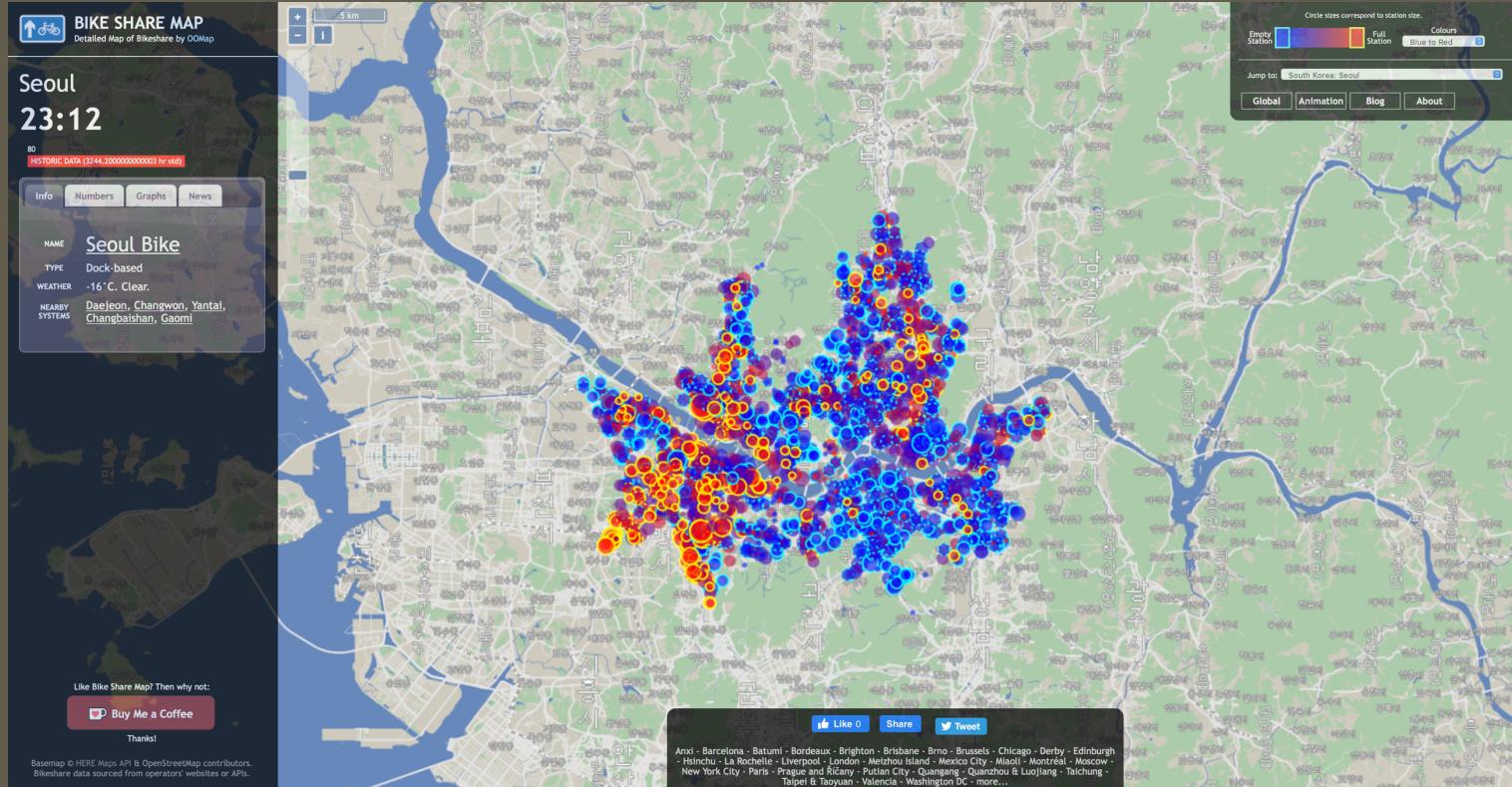


Le but est de
présenter un projet et
de détailler les étapes
qui entre dans l'étude
du dataset tel
qu'attendu par un
Data Analyst



L'exploration, la
visualisation,
l'entraînement et
l'optimisation de
modèles ainsi qu'un
API sont demandés

CONTEXTE



Map des stations de vélos et de la disponibilité en temps réel :

<https://bikesharemap.com/seoul/#/11.18067063694433/126.9894/37.5605/>

Actuellement, la location de vélos est adoptée dans plusieurs villes du monde pour faciliter les déplacements. Il est donc important de rendre cette location disponible et accessible au public au moment désiré et sans attente.

Dans certaines villes, la mise à disposition de ces vélos peut devenir un problème majeur en raison de la haute demande. Ainsi la prédition de la demande en vélos pour chaque heure de la journée est primordial pour satisfaire les habitants d'une ville.

TRAVAUX DÉJÀ RÉALISÉS



Ce dataset a déjà été l'objet
d'un article scientifique

<https://www.tandfonline.com/doi/full/10.1080/22797254.2020.1725789>



Le dataset SeoulBikeData.csv
fournit des informations
concernant les conditions
météorologiques ainsi que le
nombre de vélos loués par
heure et par date sur 2017 et
2018



Source des données
<http://data.seoul.go.kr/>

INSIDE SEOULDATABIKE.CSV

Date : année-mois-jour

Rented Bike count : Nombre de vélos loués par heure

Hour : Heure de la journée

Temperature : Température en Celsius

Humidity : Humidité en %

Windspeed : Vitesse du vent en m/s

Visibility : Visibilité en m (par 10m)

Dew point temperature : Point de rosée en Celsius

Solar radiation : Radiation solaire en MJ/m²

Rainfall : Pluie tombée en mm

Snowfall : Neige tombée en cm

Seasons : Hiver, Printemps, Été, Automne

Holiday : Vacances (Holiday/No Holiday)

Functional Day : Fonctionnement de la location (Yes/No)



On se retrouve avec un dataset de taille
8760x14

EXPLORATION ET VISUALISATION

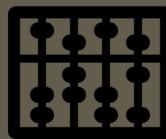
Première exploration



Pas de valeurs manquantes

On a des données qui vont du
01/12/2017 au 30/11/2018

Sur cette période on a un total
de 6 172 314 vélos loués

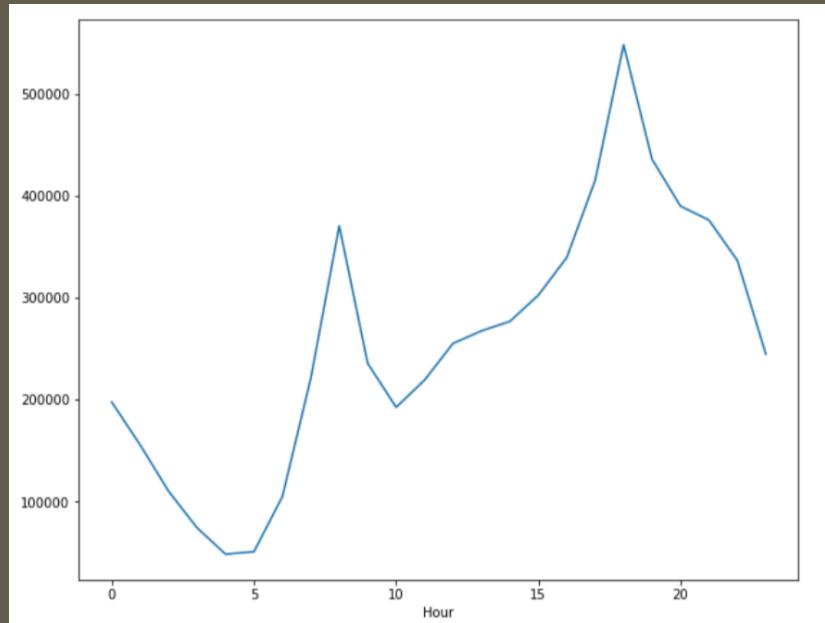


Différents types de variables
comme : object, int64 et float64

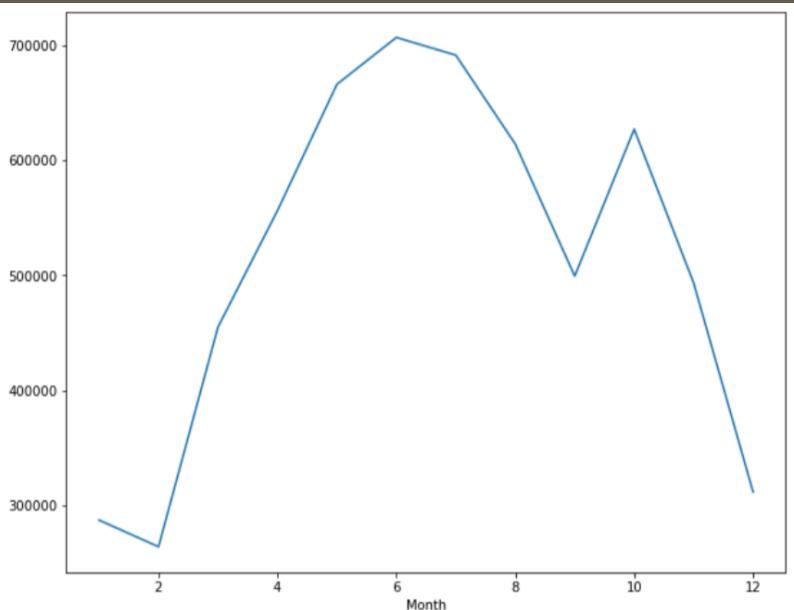


Très vite on décide de convertir
Date au format datetime et de
découper la date en Year,
Month, Day et Day_of_Week

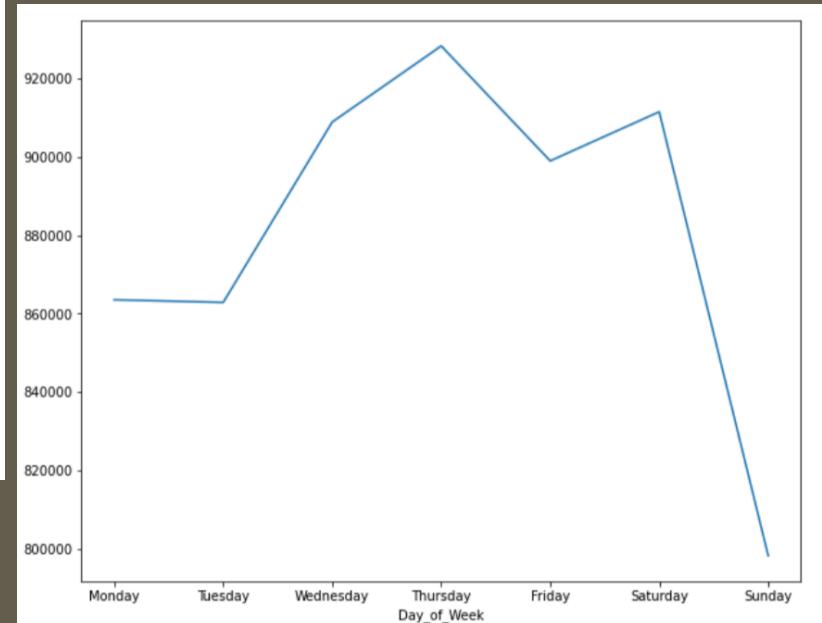
VISUALISATION D'ENSEMBLE



Nombre total de vélos loués selon l'heure de la journée



Nombre total de vélos loués selon le mois de l'année



Nombre total de vélos loués selon le jour de la semaine

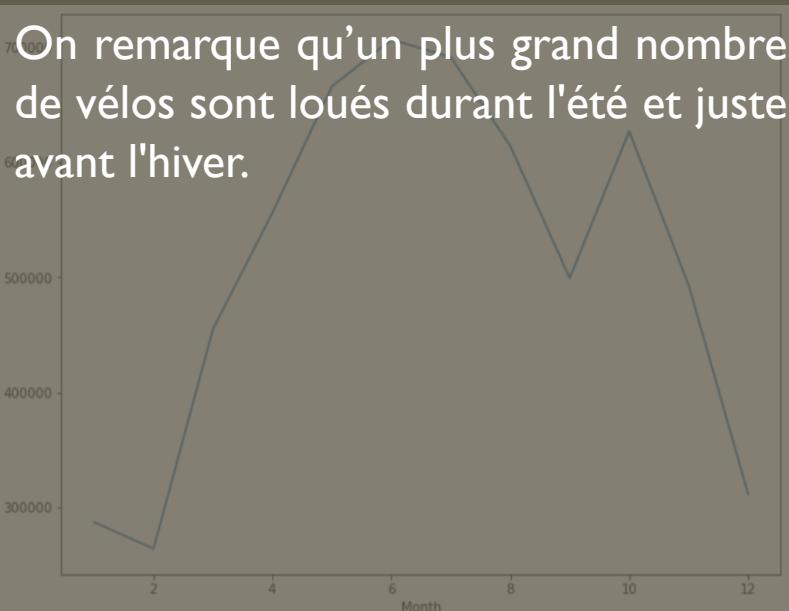
VISUALISATION D'ENSEMBLE

On remarque que les pics de location sont situés aux heures de pointes. Avec tout de même une augmentation dans l'après-midi.



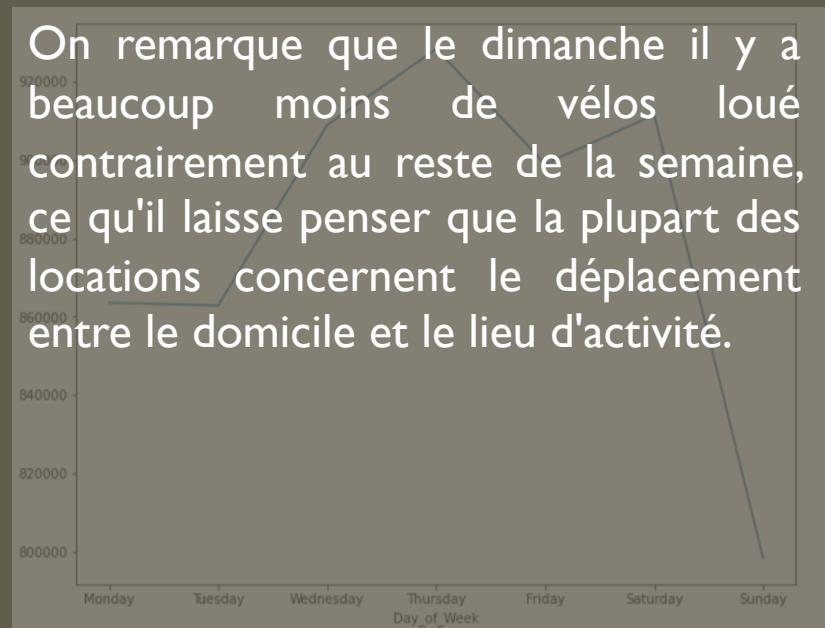
Nombre total de vélos loués selon l'heure de la journée

On remarque qu'un plus grand nombre de vélos sont loués durant l'été et juste avant l'hiver.



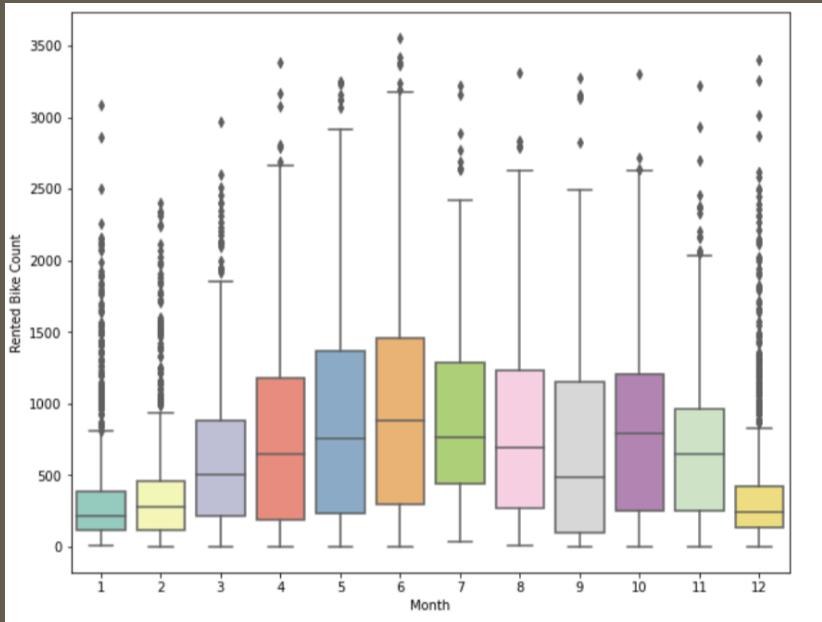
Nombre total de vélos loués selon le mois de l'année

On remarque que le dimanche il y a beaucoup moins de vélos loué contrairement au reste de la semaine, ce qu'il laisse penser que la plupart des locations concernent le déplacement entre le domicile et le lieu d'activité.

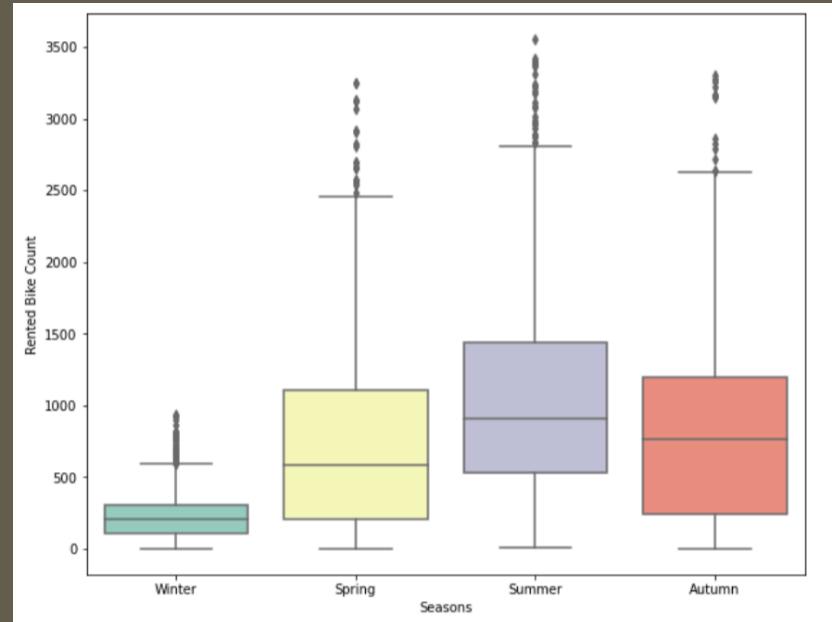


Nombre total de vélos loués selon le jour de la semaine

VISUALISATION MOIS ET SAISONS

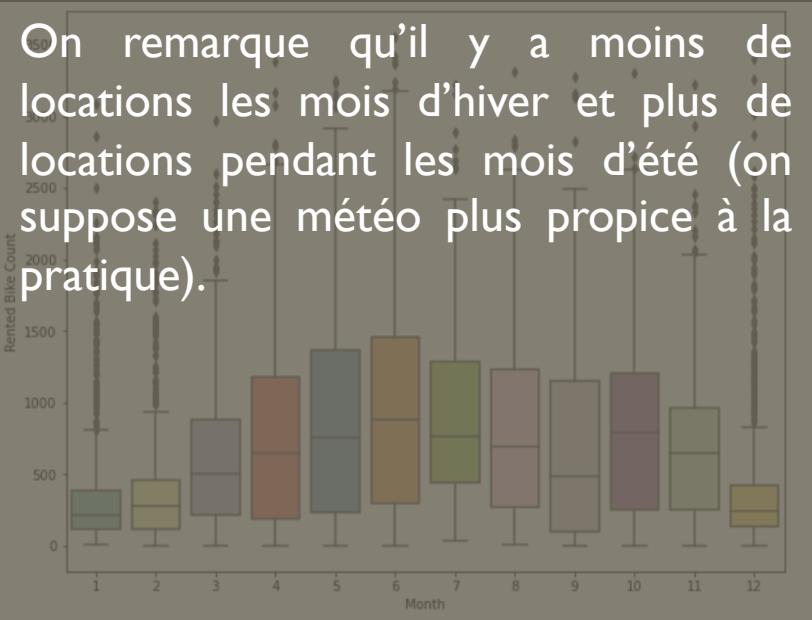


BoxPlot : Nombre de vélos loués par heure selon le mois de l'année

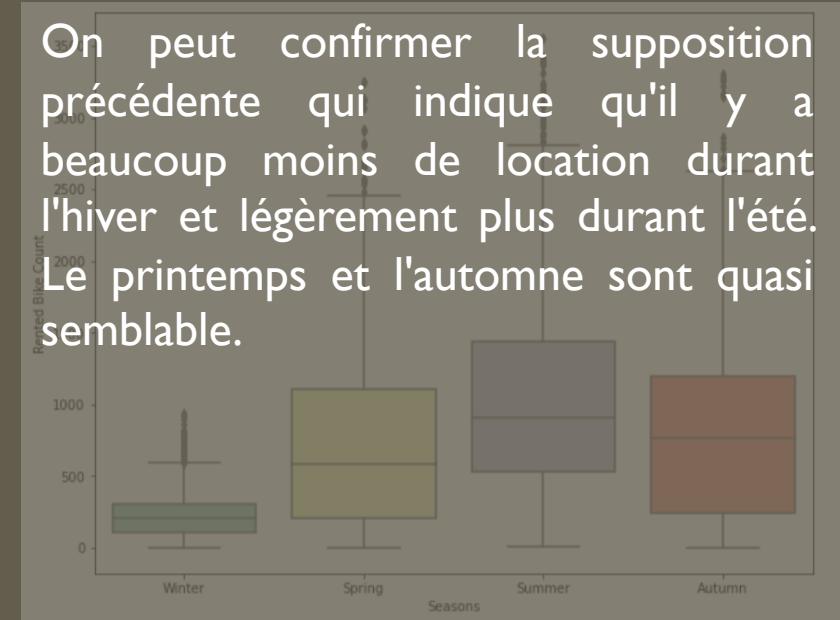


BoxPlot : Nombre de vélos loués par heure selon la saison

VISUALISATION MOIS ET SAISONS

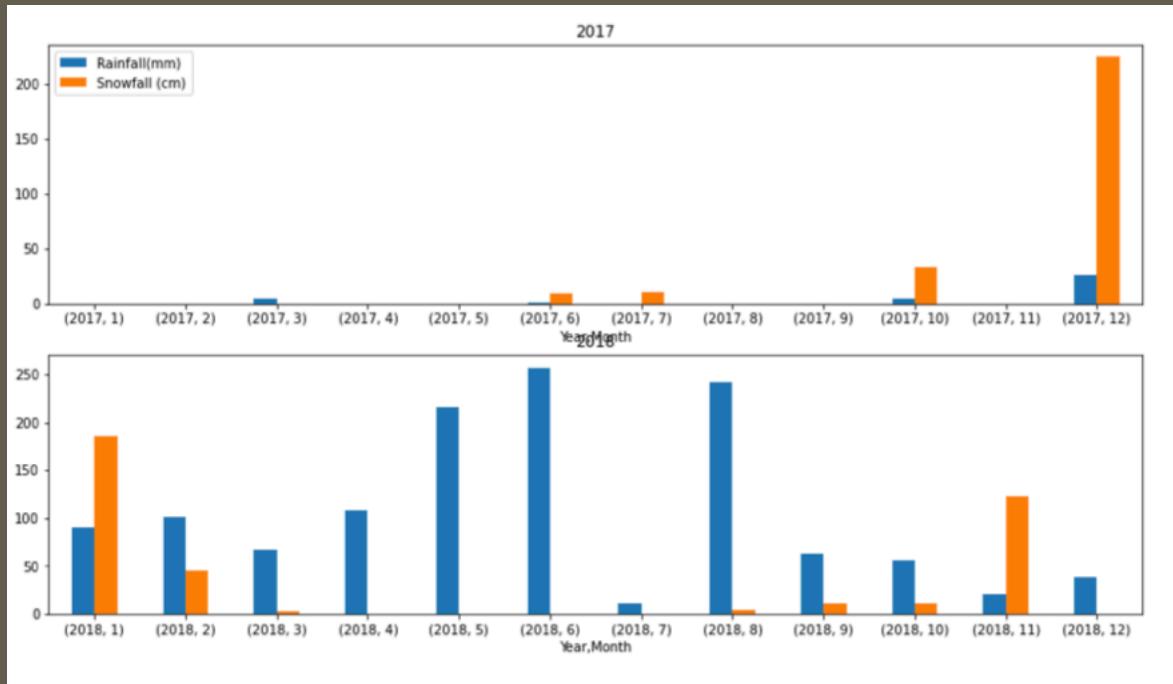


BoxPlot : Nombre de vélos loués par heure selon le mois de l'année



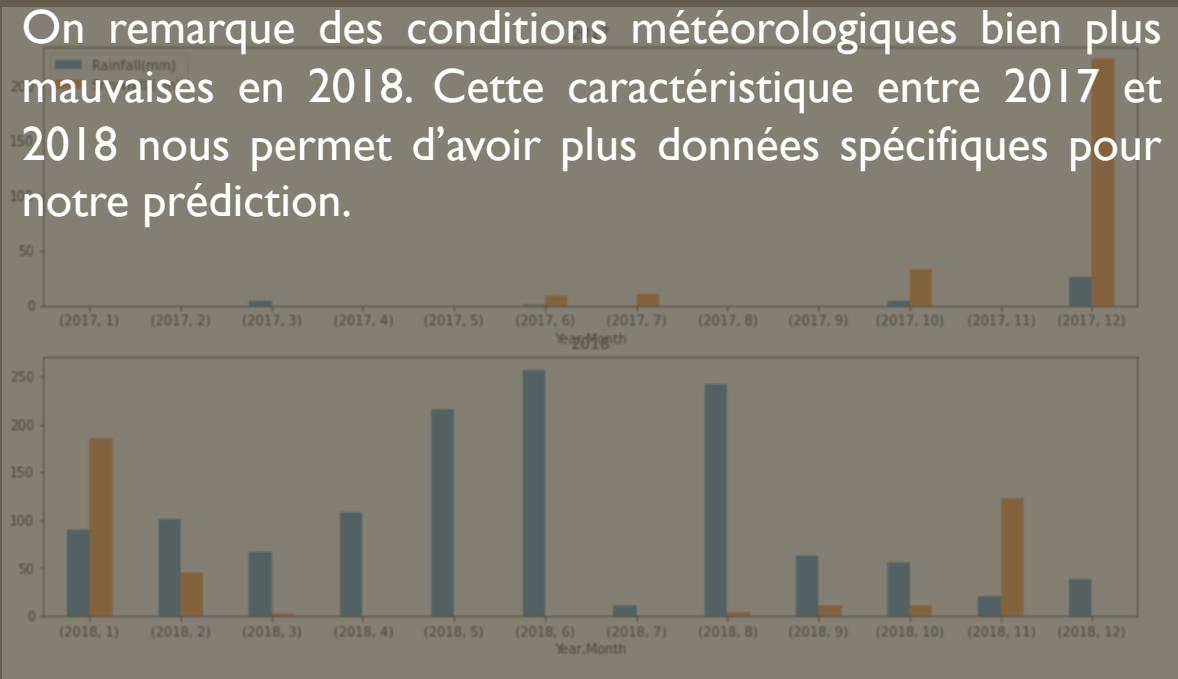
BoxPlot : Nombre de vélos loués par heure selon la saison

VISUALISATION CONDITIONS METEO



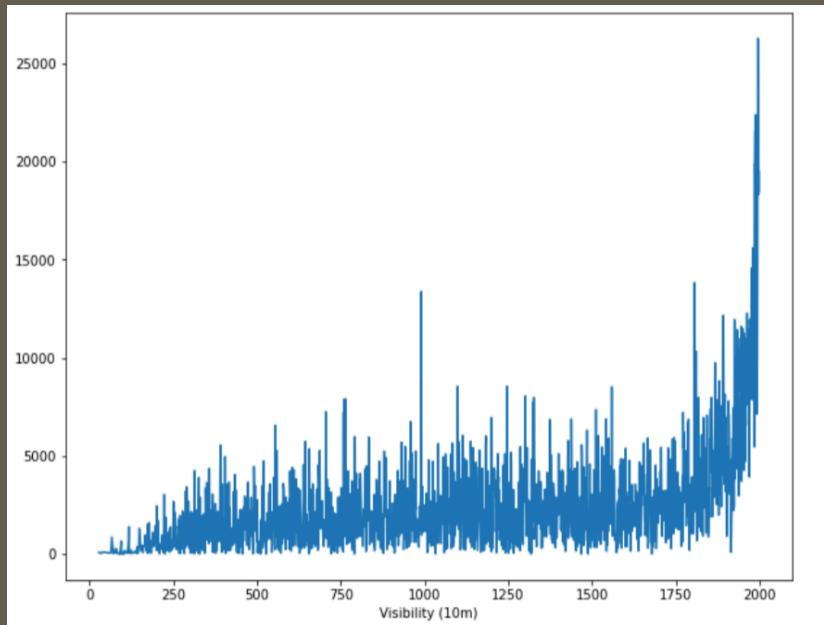
Chute de neige et de pluie selon le mois
et l'année

VISUALISATION CONDITIONS METEO

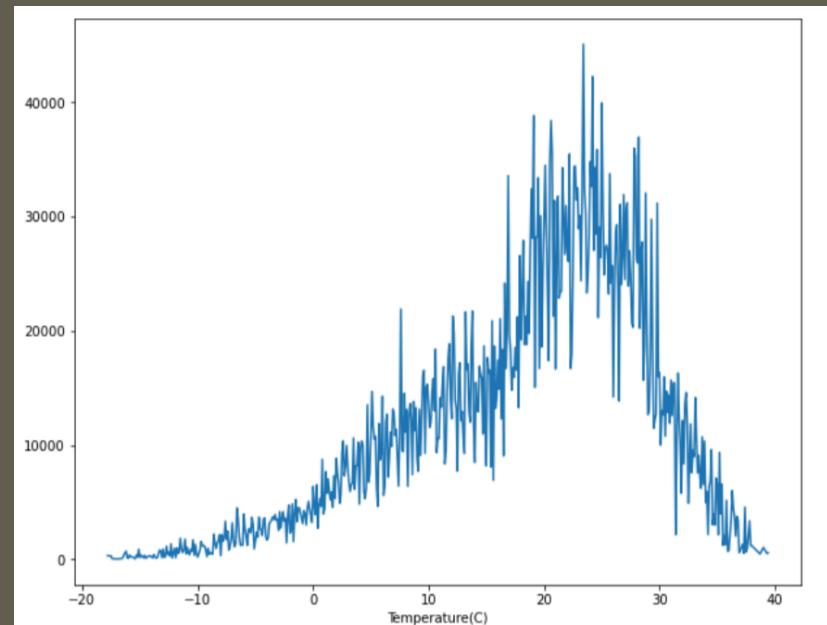


Chute de neige et de pluie selon le mois
et l'année

VISUALISATION CONDITIONS METEO

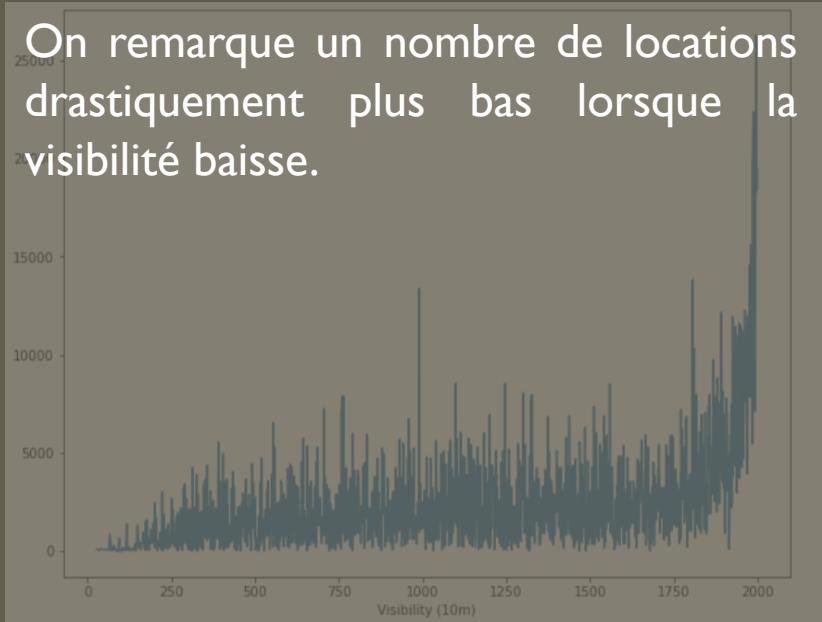


Nombre total de vélos loués selon la visibilité, inférieure à 20000 m (stade normal)

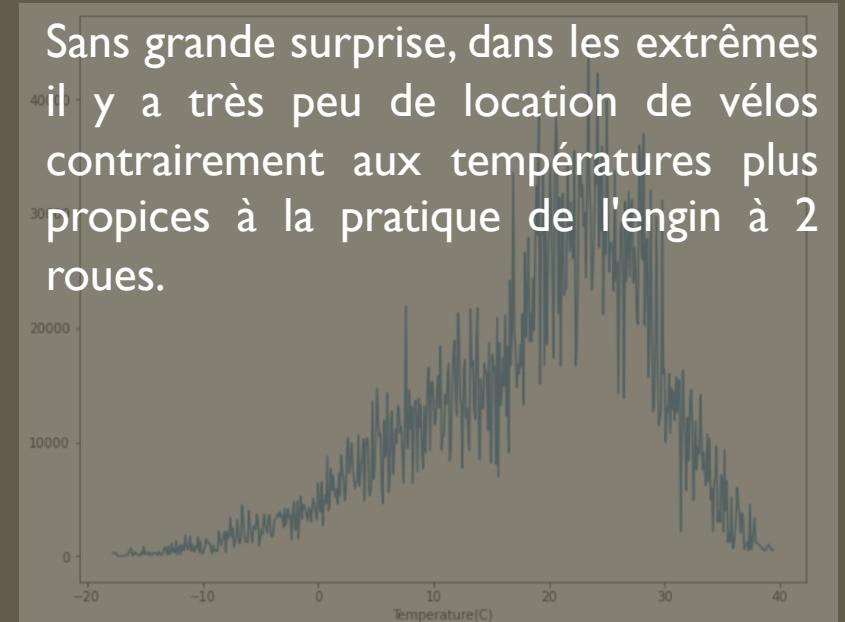


Nombre total de vélos loués selon la température

VISUALISATION CONDITIONS METEO



Nombre total de vélos loués selon la visibilité, inférieure à 20000 m (stade normal)



Nombre total de vélos loués selon la température

MÉTHODE PROPOSÉE

10

Instinctivement, notre variable cible étant le nombre de vélos loués, on peut partir sur de la prédiction d'une variable continue.



Nous allons donc partir sur un modèle de regression.
Pour pouvoir appliquer ce genre de modèle, il est nécessaire d'avoir uniquement que des valeurs numériques dans nos jeux de données.



Les étapes :

- Réaliser le pré-processing nécessaire
- Trouver le meilleur modèle de regression
- Optimiser ce modèle



PRÉ-PROCESSING



Nous avons déjà créé les variables Year, Month, Day et Day_of_Week en décomposant la variable Date



Les traitements réalisés sont :

- remplacer la variable Seasons par 1, 2, 3 ou 4 pour Hiver, Printemps, Ete, Automne
- remplacer la variable Holiday par 1 ou 0 pour Oui ou Non
- remplacer la variable Functioning Day par 1 ou 0 pour Oui ou Non
- remplacer la variable Day_of_Week par 1, 2, 3, 4, 5, 6 ou 7 pour Lundi, Mardi, Mercredi, Jeudi, Vendredi, Samedi, Dimanche
- drop la variable Date car elle est au format DateTime (non supporté dans une regression) et aussi car nous l'avons éclaté en d'autres colonnes, donc l'information n'est pas perdue.

TRAIN ET TEST SETS



On procède à la déclaration du train et du test set, pour cela il suffit d'utiliser `train_test_split` de scikit learn, en prenant comme X le dataframe pré-traité sans la variable Rented Bike Count et en prenant y comme étant la cible (Rented Bike Count).



X_{train} de taille 6132x16 & y_{train} de taille 6132
 X_{test} de taille 2628x16 & y_{test} de taille 2628

MODÈLES

Nous avons essayé différents modèles de regression et enregistrés leurs performances afin de pouvoir les comparer facilement.



Les indices d'évaluation choisis sont :

- MSE
- RMSE
- MAE
- R2



Les modèles de regression testés sont :

- Linear Regression
- Logistic Regression
- Random Forest Regressor
- Bayesian Ridge
- Gradient Boosting Regressor
- XGBoost

COMPARAISON DES MODÈLES

	Modele	MSE	RMSE	MAE	R2
5	XGBoost	37792.402516	194.402681	118.400012	0.912676
4	Gradiant Boosting Regressor	64448.354268	253.8666804	169.406657	0.851084
0	Linear Regression	200458.636045	447.726073	334.254477	0.536815
3	Bayesian Ridge	200484.608896	447.755077	334.178487	0.536755
2	Random Forest Regressor	241498.194923	491.424658	350.559485	0.441988
1	Logistic Regression	494933.458904	703.515074	501.962709	-0.143606

Pour choisir le meilleur modèle pour notre problème on regarde nos différents indices.

On cherche le modèle qui a le MSE, RMSE et MAE le plus bas ainsi que le R2 le plus haut.

Le modèle qui montre les meilleures performances est le XGBoost Regressor.

OPTIMISATION XGBOOST



Maintenant que nous avons trouvons notre modèle nous allons essayer de l'optimiser grâce à un Grid Search.

On fait varier certains hyperparamètres de notre modèle afin de selectionner les meilleurs.

```
0.9185852626465696 XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
    colsample_bynode=1, colsample_bytree=0.7, gamma=0.3, gpu_id=-1,
    importance_type='gain', interaction_constraints='',
    learning_rate=0.15, max_delta_step=0, max_depth=8,
    min_child_weight=1, missing=nan, monotone_constraints='()',
    n_estimators=100, n_jobs=0, num_parallel_tree=1, random_state=0,
    reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
    tree_method='exact', validate_parameters=1, verbosity=None)
```

COMPARAISON AVEC MODÈLES ARTICLE

A rule-based model for Seoul Bike sharing demand prediction using weather data

Sathishkumar V E & Yongyun Cho

<https://doi.org/10.1080/22797254.2020.1725789>

PUBLISHED ONLINE:
13 February 2020

Table 5 of 6

Table 5. Models performance for Seoul Bike data.



Models	Hyperparameter	R ²	Training			Testing			
			RMSE	MAE	CV	R ²	RMSE	MAE	CV
CUBIST	committees = 41 neighbours = 3	0.98	70.76	40.59	10.04	0.95	139.64	78.45	19.81
RRF	mtry = 14 coefReg = 0.505	0.98	75.83	44.80	10.76	0.93	164.85	99.05	23.39
CART	cp = 0.0001	0.92	177.00	113.45	25.12	0.87	228.94	141.37	32.49
KNN	k = 3	0.89	213.11	128.74	30.24	0.77	299.88	188.94	42.55
CIT	maxdepth = 19 mincriterion = 0.01	0.88	214.52	127.29	30.44	0.83	257.13	155.30	36.49

CSV

Performances avant optimisation :

Mean squared error : 37792.40

RMSE : 194.40

MAE : 118.40

R2 : 0.91

Performances après optimisation :

Mean squared error : 35437.49

RMSE : 188.25

MAE : 110.63

R2 : 0.92

Notre modèle produit des résultats plutôt concluants, il se situe 3^{ème} si l'on compare avec l'article.

API



API réalisée en python et flask

!

app.py

Code de l'API, on charge notre modèle et on sur les données de la requête.

Pour faire fonctionner XGBoost, nous transformons les données d'entrée en dataframe.

?

request.py

Code d'exemple de requête, on retrouve un tableau data avec 4 lignes.

Vous pouvez changer les valeurs d'entrée comme bon vous semble.

Attention néanmoins à respecter le modèle :

```
['Hour','Temperature(C)','Humidity(%)','Wind speed (m/s)','Visibility (10m)','Dew point temperature(C)','Solar Radiation (MJ/m2)','Rainfall(mm)','Snowfall (cm)','Year','Month','Day','Seasons_id','Holiday_id','FunctioningDay_id','Day_of_Week_id']
```

CONCLUSION ET OUVERTURE

Pour conclure, grâce à ce projet nous pouvons nous rendre compte de l'importance et de la valeur ajoutée d'un Data Analyst dans l'analyse et la prédition de données.

En effet, notre dataset traitait de la location de vélos à Séoul, et notre rôle était d'explorer les données et de proposer une méthode de prédition de la demande en location par heure chaque jour.

Grâce à notre exploration et nos visualisations, on peut se rendre compte rapidement que de nombreux features influence le nombre de locations. Ainsi on se rend compte de l'importance de prédire la demande afin de pouvoir répondre à un besoin utilisateur.

Après avoir testé et comparé différents modèles, nous avons choisi de continuer avec XGBoost et d'optimiser ses hyperparamètres afin d'améliorer les performances de celui-ci.

Nous obtenons donc un modèle avec des scores de : MSE : 35437.49 RMSE : 188.25 MAE : 110.63 R2 : 0.92

En ouverture, il pourrait être intéressant de travailler avec un dataset qui recense les emplacements de vélos et ainsi pouvoir prédire la demande pour chaque station de vélos (cf map slide 3).