NoSQL avec MongoDB

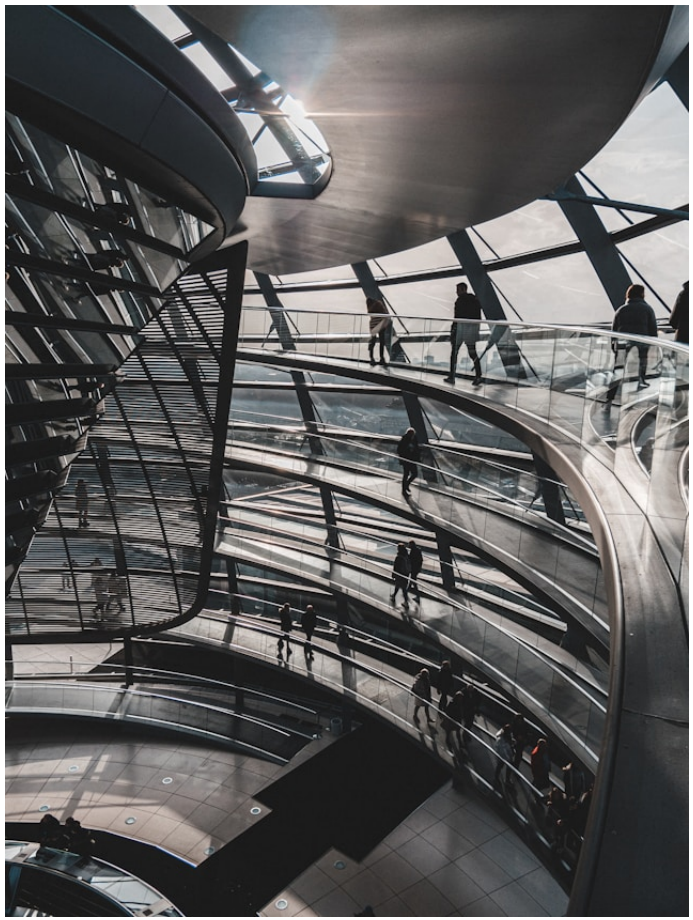
Cours 8 - Nettoyage des données

Table des matières

Utilité du nettoyage des données

Les données sont l'artéfact le **plus important** lorsqu'on entreprend un projet d'Intelligence Artificielle (AI).

Il arrive souvent que les meilleurs modèles de "Machine Learning" (ML) performent moins bien que des algorithmes de base d'Intelligence Aritificelle pour la simple raison que les données n'étaient pas de bonne qualité.



Définition et opérations populaires

Le nettoyage des données consiste à faire des opérations sur les données pour réduire ou régler les défauts existants. Les collectes de données ont **toujours un degré d'erreur associé**.

De plus, les divers modèles nécessitent des données pouvant demander des spécifications propres à respecter que des données brutes n'ont possiblement pas.

Définition et opérations populaires

Voici une liste simple* des opérations de nettoyage que nous allons voir :

- Remplacement des valeurs vides/"NaN"/"Null" par la moyenne ou des zéros.
- Transformation logarithmique (données en histogramme).
- Normalisation des données entre 0 et 1.

*il existe beaucoup plus d'opérations.

Valeurs vides

Ce nettoyage permet de remplacer les valeurs vides des attributs des documents respectifs par du contenu plus approprié lors de l'entraînement de modèles.

La valeur moyenne est généralement utilisée, puisqu'elle permet de garder l'uniformité des données d'une manière simple. Dépendamment du domaine d'affaires, une autre règle pourrait être choisie (i.e. zéros, maximum, minimum, etc.).

Valeurs vides

Listing: https://medium.com/@stevencibambo/loading-data-from-mongodb-with-pymongo-for-analysis-c0f61a8538a0

```
1  # Steve Levesque, All rights reserved where applicable
2  import pandas as pd
3  from pymongo import MongoClient
4
5  client = MongoClient()
6  db = client.VegetableJSONSchema
7  collection = db.VegetablesValidated
8  documents = collection.find()
9
10 df = pd.DataFrame(documents)
11
12 # Handle missing values, using the mean for numerical is recommended.
13 value = "protein (g)"
14 df[value] = df[value].fillna(int(df[value].mean()))
15 df = df.fillna(0)
16
17 print(df.iloc[0:2].to_string())
18 document_count = collection.count_documents({})
19 dataframe_count = len(df.index)
20 print(document_count, dataframe_count)
```

Transformation logarithmique

Ce nettoyage permet de rendre les données uniformes en termes de distribution des entrées, permettant généralement de rendre les modèles statistiques plus performants.

Transformation logarithmique

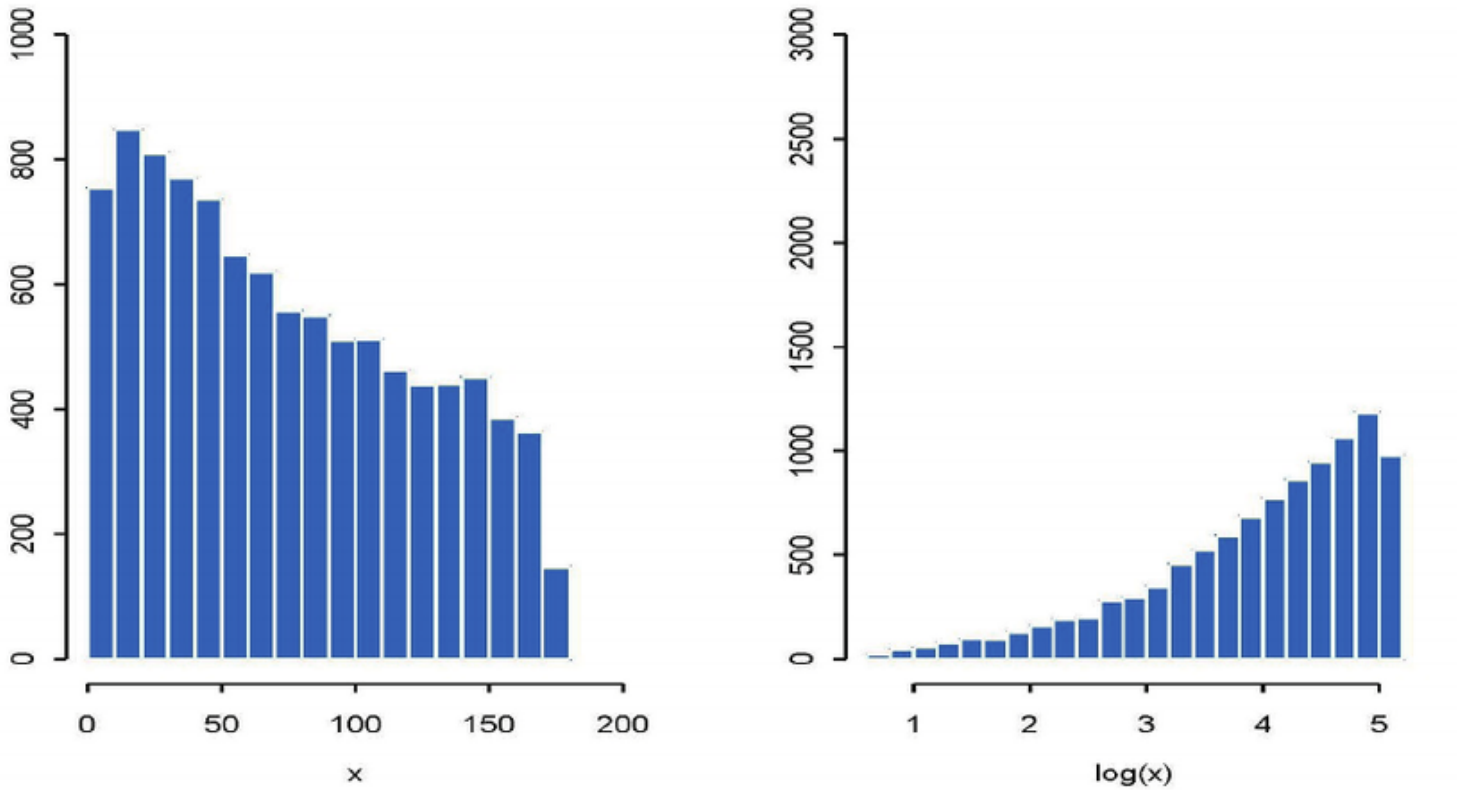


Figure: https://www.researchgate.net/figure/Histograms-of-original-data-left-plot-and-log-transformed-data-right-plot-from-a_fig1_264500976

Transformation logarithmique

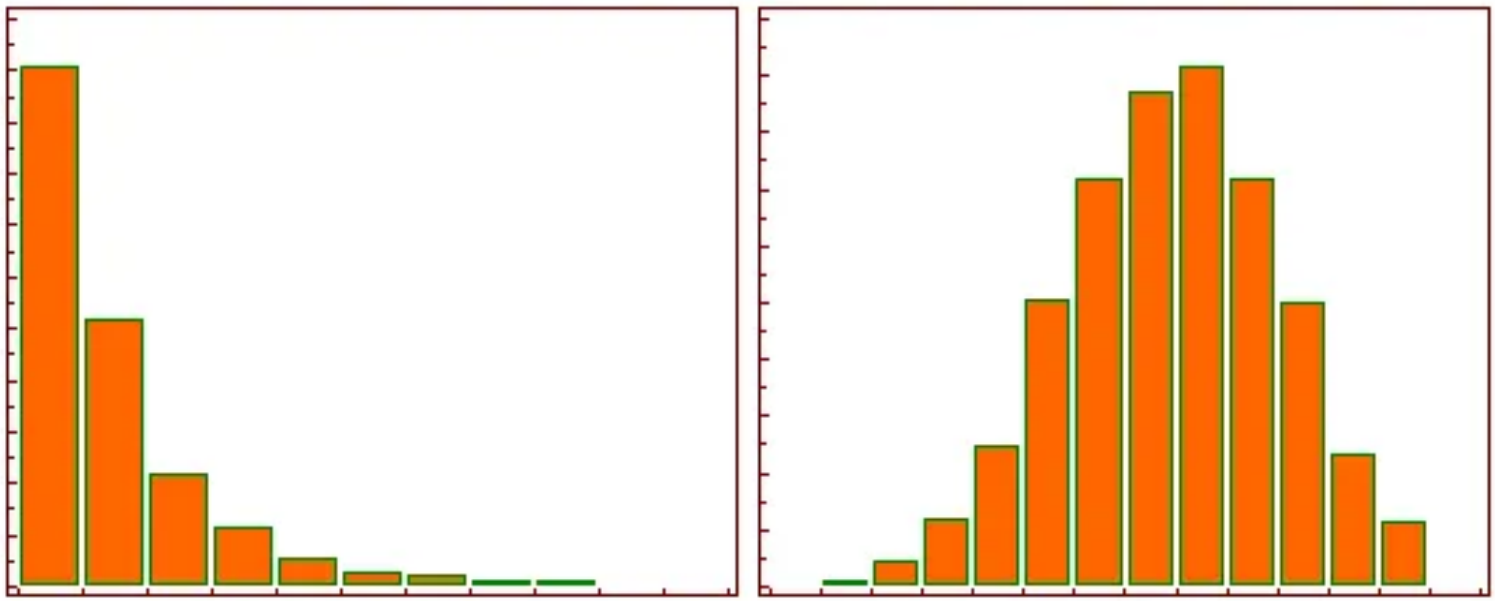


Figure:
https://medium.com/@kyawsawhtoon/log-transformation-purpose-and-interpretation-9444b4b049c9

Transformation logarithmique

Listing: https://medium.com/@stevencibambo/loading-data-from-mongodb-with-pymongo-for-analysis-c0f61a8538a0

```
1  # Steve Levesque, All rights reserved where applicable
2  import matplotlib.pyplot as plt, pandas as pd, numpy as np
3  from pymongo import MongoClient
4
5  column = 'protein (g)'
6  client = MongoClient()
7  db = client.admin
8  collection = db.Vegetables
9
10 df_ori = pd.DataFrame(collection.find())
11 data = np.array([document[column] for document in collection.find()])
12
13 # Apply logarithmic transformation
14 transformed_data = np.log(data)
15 df_new = pd.DataFrame(transformed_data)
16
17 df_ori.hist()
18 df_new.hist()
19 plt.show()
```

Normalisation

Ce nettoyage permet aux modèles de mieux performer puisque les données sont uniformément rapprochées dans une étendue entre 0 et 1.

Par exemple, la disparité (distance) entre l'âge et le salaire est grande au niveau numérique. Les modèles, surtout ceux qui sont sensibles à la notion de distance, sont fortement influencés par une telle distribution des données.

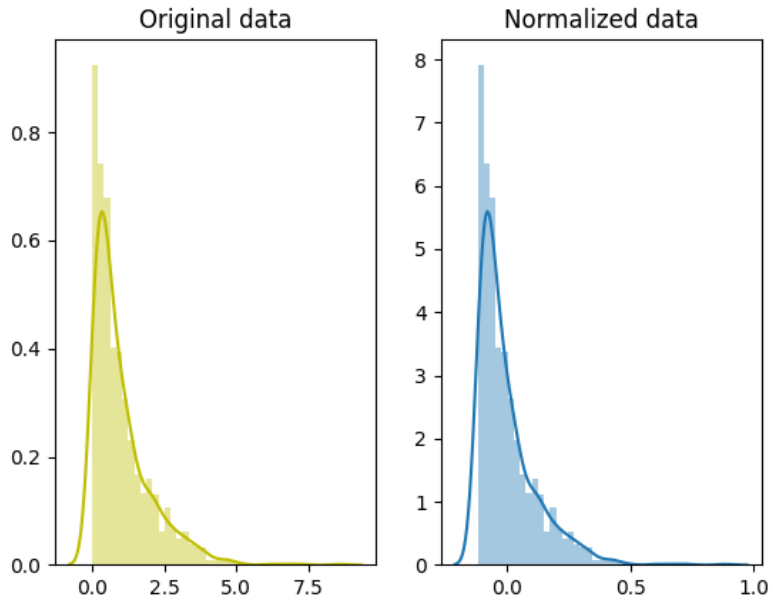


Figure: La topologie des données reste la même.

Normalisation

Listing: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

```
1  # Steve Levesque, All rights reserved where applicable
2  import pandas as pd
3  from sklearn.preprocessing import MinMaxScaler
4  from pymongo import MongoClient
5
6  client = MongoClient()
7  db = client.admin
8  collection = db.Vegetables
9  documents = collection.find()
10
11 df = pd.DataFrame(documents)
12
13 # Normalize numeric fields
14 scaler = MinMaxScaler()
15 w_g = 'water (g)'
16 p_g = 'protein (g)'
17 df[[w_g, p_g]] = scaler.fit_transform(df[[w_g, p_g]])
18
19 print(df[[w_g, p_g]])
20 print(df[[w_g, p_g]].min())
21 print(df[[w_g, p_g]].max())
```

Bibliographie

- https://medium.com/@stevencibambo/loading-data-from-mongodb-with-pymongo-for-analysis-c0
- https://medium.com/@kyawsawhtoon/log-transformation-purpose-and-interpretation-9444b4b0
- https://www.researchgate.net/figure/Histograms-of-original-data-left-plot-and-log-transfor fig1_264500976
- https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html