

Prompt documentation: step 1: data collection and cleaning

```
User
détaille moi plus l'exploration et la préparation des données

ChatGPT
### **Exploration et Préparation des Données en Détail**

L'exploration et la préparation des données sont des étapes essentielles pour garantir un bon entraînement du modèle de machine learning. Voici un plan détaillé pour analyser, nettoyer et transformer vos données avant de les utiliser dans un modèle.

---

## **1. Charger et Comprendre les Données**

### ✅ **Charger le dataset**
Ouvrez un notebook Python (`.eda.ipynb`) et chargez les données :



```
python
import pandas as pd

Charger le dataset
df = pd.read_csv("obesityDataset_raw_and_data_synthetic.csv")

Afficher les premières lignes
df.head()
```



---

### ✅ **Comprendre la structure des données**
- Nombre de lignes et de colonnes :


```
python
df.shape
```


- Types de données :


```
python
df.info()
```


- Statistiques générales :


```
python
df.describe()
```



---

Cela vous aidera à repérer les valeurs extrêmes et à comprendre les distributions.
```

```
## **2. Vérification des Valeurs Manquantes**
Les valeurs manquantes peuvent causer des problèmes dans les modèles. Vérifiez leur présence :



```
python
df.isnull().sum()
```



Si des valeurs sont manquantes, plusieurs stratégies existent :
- Supprimer les lignes (`.dropna()`)
- Remplir les valeurs manquantes avec la moyenne (`.fillna(df.mean())`)
- Utiliser une imputation avancée (KNN, interpolation)

Si les valeurs manquantes sont nombreuses, documentez la stratégie choisie.

---

## **3. Identification et Gestion des Valeurs Aberrantes (Outliers)**
Les valeurs aberrantes peuvent fausser les modèles. Voici comment les détecter :

### ✅ **1. Visualiser les distributions avec des boxplots**


```
python
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(15, 8))
sns.boxplot(data=df)
plt.xticks(rotation=90) # Rotation des labels
plt.show()
```



Les points en dehors des moustaches du boxplot sont des valeurs aberrantes.

### ✅ **2. Supprimer ou corriger les outliers**
Deux méthodes :
- Supprimer les valeurs trop extrêmes :


```
python
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1

df = df[~((df < (Q1 - 1.5 * IQR)) | (df > (Q3 + 1.5 * IQR))).any(axis=1)]
```


- Transformer les valeurs aberrantes (log, min-max scaling)
```

```
User
est ce qu'il n'y a pas de méthodes pour déterminer les outliers objet?

ChatGPT
### ❖ Détection des Outliers dans les Variables Catégoriques (Objet)**
Contrairement aux variables numériques où l'on utilise des méthodes statistiques (IQR, Z-score, etc.), la détection des outliers dans les variables catégoriques (`.object`) repose sur d'autres approches.

---

## **1. Méthode 1 : Détection des Valeurs Rares (Fréquence d'apparition)**
Une valeur catégorique peut être considérée comme un outlier si elle apparaît très rarement.

### ✅ Code pour détecter les valeurs rares

```
python
for col in df.select_dtypes(include=['object']).columns: # Parcours des colonnes catégoriques
 print(f"Valeurs uniques dans '{col}':")
 print(df[col].value_counts()) # Nombre d'occurrences pour chaque valeur
 print("\n")

Interprétation :
- Si une catégorie a très peu d'occurrences (ex. moins de 1% des données), elle peut être un outlier catégorique.
- Par exemple, si une colonne "Moyen de transport" contient 99% "Voiture" et 1% "Trottinette", "Trottinette" pourrait être un outlier.

✅ Gérer les valeurs rares
Si vous détectez une valeur rare, vous pouvez :
1. La regrouper avec une autre catégorie (`.Autre` ou `Inconnu`) :


```
python
df['MTRANS'] = df['MTRANS'].replace(['Trottinette'], 'Autre')
```


2. L'éliminer si elle est trop rare :


```
python
df = df[df['MTRANS'] != 'Trottinette']
```

```


```

User

est ce que ces codes modifient directement la dataset?

ChatGPT

Oui et non, cela dépend de la manière dont le code est écrit. Analysons chaque méthode pour comprendre si elle **modifie directement** le dataset ou non.

```
---
## **💡 Méthode 1 : Suppression des outliers**
...python
df_cleaned = df[~((df < lower_bound) | (df > upper_bound)).any(axis=1)]
...
* **Ne modifie pas le dataset original** `df`, mais crée un nouveau dataset `df_cleaned`.
* Si vous voulez modifier directement `df`, vous devez faire :
...python
df = df[~((df < lower_bound) | (df > upper_bound)).any(axis=1)]
...
💡 Attention : Cette suppression est irréversible, sauf si vous gardez une copie du dataset original.
---
## **💡 Méthode 2 : Winsorization**
...python
df["Weight"] = winsorize(df["Weight"], limits=[0.05, 0.05])
df["Age"] = winsorize(df["Age"], limits=[0.05, 0.05])
...
* Modifie directement `df`, donc les valeurs extrêmes sont remplacées dans le dataset original.
* Si vous voulez éviter la modification directe, faites une copie :
...python
df_copy = df.copy()
df_copy["Weight"] = winsorize(df_copy["Weight"], limits=[0.05, 0.05])
df_copy["Age"] = winsorize(df_copy["Age"], limits=[0.05, 0.05])
...
Cela permet de tester les modifications sans affecter le dataset original.
---
```

User

ou je vais trouver la copie?

ChatGPT

Lorsque vous faites une copie du dataset en Python avec `df.copy()`, elle est **stockée en mémoire**, mais **pas enregistrée automatiquement** sur votre ordinateur. Voici comment la retrouver et l'utiliser.

```
---
## **💡 Où se trouve la copie du dataset ?**
Si vous créez une copie comme ceci :
...python
df_copy = df.copy()
...
- `df_copy` existe temporairement en mémoire RAM tant que le programme est en cours d'exécution.
- Il ne sera pas enregistré sur votre disque dur, sauf si vous l'enregistrez manuellement.
---
## **💡 Comment afficher et vérifier la copie ?**
Vous pouvez voir si la copie existe en affichant ses premières lignes :
...python
print(df_copy.head()) # Affiche les premières lignes
print(df_copy.shape) # Vérifie le nombre de lignes et colonnes
...
Si `df_copy` est bien défini, ces commandes fonctionneront.
---
## **💡 Comment enregistrer la copie sur votre ordinateur ?**
Si vous voulez garder la copie même après avoir fermé le programme, vous devez l'enregistrer sous forme de fichier CSV ou Excel.
### **💡 Sauvegarde en CSV**
...python
df_copy.to_csv("copie_du_dataset.csv", index=False)
...
- 💡 Le fichier "copie_du_dataset.csv" sera enregistré dans le même dossier que votre script Python.
- 💡 Pour enregistrer à un endroit spécifique :
```

User

[File]: [file-TuCARTPE2ZnnPs8b65RomG-image.png](#)
voici la distribution de mon data set décrits la moi

ChatGPT

💡 Analyse de la Distribution des Classes dans votre Dataset
L'image montre une répartition des **niveaux d'obésité** dans votre dataset.

```
---
### **💡 Statistiques Générales**
| **Statistique** | **Valeur** |
|-----|-----|
| Nombre total d'échantillons (Count) | 1000 |
| Nombre d'erreurs (Error) | 0 (aucune erreur détectée) |
| Nombre de valeurs manquantes (Empty) | 0 |
| Nombre de classes distinctes (Distinct) | 7 |
| Nombre de valeurs uniques (Unique) | 0 (chaque classe a plusieurs occurrences) |
| Nombre de chaînes vides (Empty String) | 0 |
---
💡 Conclusion : Aucune valeur manquante ni erreur détectée, ce qui signifie que votre dataset est propre et bien formaté.
---
### **💡 Répartition des Classes**
Votre variable cible (NOBeyesdad) contient 7 catégories, représentant différents niveaux d'obésité et de poids :
| **Classe** | **Quantité (approximative)** | **Observation** |
|-----|-----|-----|
| Overweight Level I | Majoritaire | La classe la plus représentée |
| Normal weight | Majoritaire | Deuxième classe la plus fréquente |
| Insufficient weight | Majoritaire | Troisième classe en importance |
| Overweight Level II | Modérée | Moins représentée que les trois premières |
| Obesity Type I | Minoritaire | Faible présence dans le dataset |
| Obesity Type III | Très minoritaire | Rarement présente |
| Obesity Type II | Très minoritaire | La classe la moins représentée |
---
💡 Interprétation :
- Les catégories les plus présentes sont "Overweight Level I", "Normal weight" et "Insufficient weight".
- Les catégories "Obesity Type II" et "Obesity Type III" sont sous-représentées, ce qui indique un déséquilibre des classes (Class Imbalance).
```

User

par quoi je dois commencer déterminer le modèle ou choisir la méthode pour remédier à l'imbalance des classes?

ChatGPT

💡 Par quoi commencer : Choisir le Modèle ou Traiter l'Imbalance des Classes ?
La meilleure approche est **de tester d'abord un modèle sans corriger l'imbalance**, puis d'analyser les performances pour décider **si l'imbalance affecte les résultats**.
Voici une méthode en **3 étapes** :

1️⃣ Étape 1 : Tester un Modèle Baseline sans Modifier l'Imbalance

Pourquoi ?

- Cela permet de voir **si l'imbalance des classes impacte vraiment la performance** du modèle.
- Certains modèles comme **Random Forest, XGBoost, LightGBM** gèrent déjà bien l'imbalance **avec leurs paramètres internes**.

💡 Actions à Faire

1. **Diviser les données en Train/Test** :

```
python
from sklearn.model_selection import train_test_split

X = df.drop(columns=["Nbeyesdad"]) # Features
y = df["Nbeyesdad"] # Target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y, random_state=42)
```

💡 **Pourquoi `stratify=y` ?** Cela garde le même ratio de classes dans Train/Test.

2. **Entraîner un Modèle Simple (XGBoost)**

```
python
from xgboost import XGBClassifier
from sklearn.metrics import classification_report

model = XGBClassifier(use_label_encoder=False, eval_metric="mlogloss") # Modèle sans gestion spéciale de l'imbalance
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print(classification_report(y_test, y_pred)) # Vérifier les performances
```