



# *Projet analyse données*

Analyse des ventes

# Objectifs de l'analyse des ventes

**Optimiser les performances commerciales :** Déceler les tendances, les modèles récurrents et les opportunités afin d'accroître les ventes et d'améliorer l'efficacité opérationnelle.

**Anticiper les ventes à venir :** Prédire les volumes de ventes futurs en se basant sur des données et des tendances historiques pour mieux planifier les stocks et les ressources.

**Déterminer les facteurs influents :** Identifier les éléments clés (tels que les caractéristiques des produits, les conditions du marché ou les campagnes promotionnelles) qui impactent les résultats des ventes.

**Faciliter la prise de décisions stratégiques :** Fournir des analyses précises pour orienter les choix en matière de tarification, de marketing et de gestion des stocks.

**Augmenter la rentabilité :** Développer des stratégies de vente optimales en mettant l'accent sur les produits ou segments de clientèle les plus profitables.

**Atteindre les objectifs commerciaux :** Suivre et ajuster les stratégies de vente afin de garantir l'atteinte des cibles fixées par l'entreprise.

# L'ensemble de données

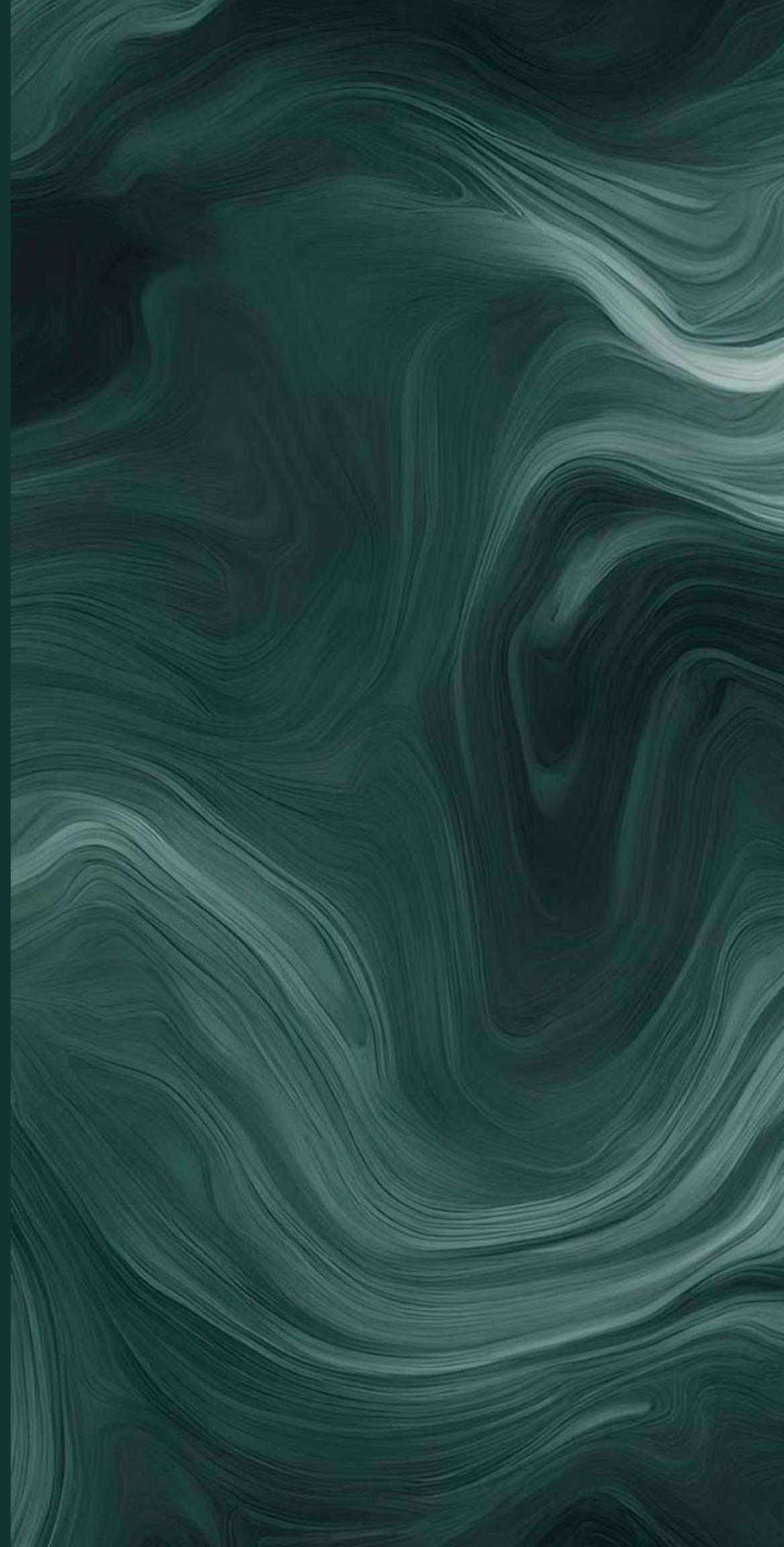
L'ensemble de données statistiques utilisé dans cette étude provient d'une plateforme de E-Commerce électronique, qui comptabilise les données de vente de divers produits vestimentaires du 1er mai 2016 au 1er avril 2019. L'ensemble de données statistiques comprend les ventes et différentes fonctionnalités qui peuvent refléter les ventes à savoir :

- **(S ; unité : milliard de Yuan)** : les ventes
- **(X4 ; unité :milliards)** : pages vues
- **(X8 ; unité : 10 millions)** : nombre d'achats supplémentaires
- **(X9 ; unité : 100 mille)** : groupe de clients Indice
- **(X10 ; unité : million)** : indice de transaction.

# Processus d'analyse à réaliser



# 1—Collecte des données



# Collecte des données

Pour commencer, On va charger le fichier Excel et examiner les différentes feuilles pour comprendre les données disponibles. Cela nous permettra de définir un projet pertinent à partir des informations contenues dans le fichier. On va exécuter le code pour charger et afficher un aperçu des données.

```
import pandas as pd

# Charger toutes les feuilles d'un fichier Excel dans un dictionnaire
def read_all_sheets_from_excel(filepath):
    excel_data = pd.ExcelFile(filepath)
    dataframes = {sheet_name: excel_data.parse(sheet_name) for sheet_name in excel_data.sheet_names}
    return dataframes

# Charger le fichier Excel et afficher un aperçu des différentes feuilles
FILEPATH = 'DataSetVentes.xlsx'
dataframes = read_all_sheets_from_excel(FILEPATH)

# Afficher les noms des feuilles et un aperçu des 15 premières lignes de chaque feuille
for k, v in dataframes.items():
    print('dataframe: ' + k)
    print(v.head(3))
```

# Collecte des données

Le résultat du code:

```
dataframe: Pants sales
    Date  S/ billion Yuan  X4/ billion  X8/ 10\nmillion  X9/ 100 thousand \
0 2016-05-01           2.23          3.84          1.48          1.12
1 2016-06-01           2.14          3.85          1.44          1.09
2 2016-07-01           1.72          3.30          1.19          0.97

   X10/ million
0           1.19
1           1.16
2           1.03

dataframe: Dress sales.
    Date  S/ billion Yuan  X4/ billion  X8/ 10\nmillion  X9/ 100 thousand \
0 2016-05-01           5.29          11.38          3.03          1.21
1 2016-06-01           5.66          12.45          3.21          1.24
2 2016-07-01           4.87          11.70          2.89          1.18

   X10/ million
0           1.98
1           2.06
2           1.88

dataframe: Sweater sales
    Date  S/ billion Yuan  X4/ billion  X8/ 10\nmillion  X9/ 100 thousand \
0 2016-05-01            0.63          1.11          3.66          4.71
1 2016-06-01            0.46          0.82          2.72          3.91
2 2016-07-01            0.42          0.81          2.70          3.55
...
   X10/ million
0           5.87
1           4.97
2           4.69
```

# Collecte des données

On voit que le fichier contient trois feuilles de données de ventes différentes :

- Ventes de pantalons (Pants sales)
- Ventes de robes (Dress sales)
- Ventes de pulls (Sweater sales)

Chaque feuille contient des données sur les ventes en différentes unités (milliards de Yuan, millions, etc.) pour différentes dates.

## 1 S (Ventes, unité : milliard de Yuan)

- Cette variable représente les ventes totales réalisées sur la plateforme pour les produits vestimentaires, exprimées en **milliards de Yuan**.
- C'est la **variable cible principale**, car l'objectif est de comprendre et d'améliorer les performances des ventes.

## 2 X4 (Pages vues, unité : milliards)

- Indique le nombre total de pages consultées sur la plateforme pendant la période donnée, exprimé en **milliards**.
- Cette variable est un indicateur clé de l'intérêt ou de l'engagement des clients envers les produits et la plateforme.

## 3 X8 (Nombre d'achats supplémentaires, unité : 10 millions)

- Correspond au nombre de fois où les clients ont ajouté des articles supplémentaires à leurs paniers avant de finaliser leurs commandes, exprimé en **dizaines de millions**.
- Cela reflète le comportement d'achat et la propension des clients à acheter plusieurs articles lors d'une transaction.

## 4 X9 (Indice de groupe de clients, unité : 100 mille)

- Cette variable représente un **indice agrégé** qui reflète le comportement ou la segmentation des groupes de clients, exprimé en **centaines de milliers**.
- Elle peut inclure des données telles que le profil des clients, leurs préférences ou leur fidélité envers la plateforme.

## 5 X10 (Indice de transaction, unité : million)

- Représente un **indice global des transactions** réalisées sur la plateforme, exprimé en **millions**.
- Cet indice est lié au volume et à la fréquence des transactions et peut inclure des données sur les promotions ou les périodes de forte activité (ex. : soldes).

## 2—Echantillonnage



# Echantillonnage

On va appliquer une méthodes d'échantillonnage sur nos données de ventes.

## Échantillonnage aléatoire simple

**Description** : Chaque élément de la population a une probabilité égale d'être sélectionné.

**Avantages** : Facile à mettre en œuvre ; garantit que l'échantillon est représentatif si la taille est suffisante.

**Inconvénients** : Nécessite une liste complète de la population ; peut-être coûteux pour les grandes populations.

L'échantillonnage aléatoire simple assure que chaque vente, indépendamment du type de produit, a une probabilité égale d'être incluse dans l'échantillon. On peut directement tirer un échantillon de chaque feuille ou des trois combinées.

# Echantillonnage

Le code extrait des échantillons aléatoires représentant 60% des données pour chaque catégorie (pantalons, robes et pulls) à partir d'un fichier Excel. Cette méthode permet de travailler sur un sous-ensemble représentatif des données, facilitant ainsi des analyses plus rapides et ciblées. La taille de l'échantillon a été choisie en raison de la taille relativement petite des données disponibles. En sélectionnant 60%, on obtient un échantillon suffisamment important pour réaliser des analyses significatives tout en simplifiant les calculs.

```
# Lire les données
df_pants = pd.read_excel('DataSetVentes.xlsx', sheet_name='Pants sales')
df_dress = pd.read_excel('DataSetVentes.xlsx', sheet_name='Dress sales.')
df_sweater = pd.read_excel('DataSetVentes.xlsx', sheet_name='Sweater sales')

# Échantillonnage aléatoire simple (60% des données)
sample_size = 0.6
pants_sample = df_pants.sample(frac=sample_size, random_state=42)
dress_sample = df_dress.sample(frac=sample_size, random_state=42)
sweater_sample = df_sweater.sample(frac=sample_size, random_state=42)
# Réinitialiser les indices des échantillons
pants_sample = pants_sample.reset_index(drop=True)
dress_sample = dress_sample.reset_index(drop=True)
sweater_sample = sweater_sample.reset_index(drop=True)

print("Taille des échantillons (60% des données):")
print("Pantalons:", len(pants_sample), "sur", len(df_pants), "observations")
print("Robes:", len(dress_sample), "sur", len(df_dress), "observations")
print("Pulls:", len(sweater_sample), "sur", len(df_sweater), "observations")
```

Le résultat du code:

```
Taille des échantillons (60% des données):
Pantalons: 22 sur 36 observations
Robes: 22 sur 37 observations
Pulls: 22 sur 36 observations
```

### 3—Suppression des doublons



# Suppression des doublons

La suppression des doublons après l'échantillonnage est cruciale pour garantir la fiabilité des données. Elle évite les biais liés à la présence de données identiques dans l'échantillon, assurant ainsi que les résultats de l'analyse reflètent fidèlement la population d'origine. En plus de garantir l'intégrité des données, la suppression des doublons améliore l'efficacité des algorithmes et optimise l'utilisation de l'espace de stockage.

```
# Suppression des doublons dans chaque dataset
def remove_duplicates(df, name):
    initial_count = len(df)
    df_cleaned = df.drop_duplicates()
    final_count = len(df_cleaned)
    print("Nombre de doublons supprimés dans", name, ":", initial_count - final_count)
    return df_cleaned

# Nettoyage des données
df_pants_cleaned = remove_duplicates(pants_sample, "Pantalons")
df_dress_cleaned = remove_duplicates(dress_sample, "Robes")
df_sweater_cleaned = remove_duplicates(sweater_sample, "Pulls")

# Afficher un aperçu des données nettoyées
print("Aperçu des données nettoyées (Pantalons):")
print(df_pants_cleaned.head())
```

Le résultat du code:

```
Nombre de doublons supprimés dans Pantalons : 0
Nombre de doublons supprimés dans Robes : 0
Nombre de doublons supprimés dans Pulls : 0
Aperçu des données nettoyées (Pantalons):
      Date  S/ billion Yuan  X4/ billion  X8/ 10\nmillion  X9/ 100 thousand \
0  2019-04-01           3.12          5.45          1.95          1.04
1  2017-06-01           2.78          6.62          1.92          1.18
2  2018-07-01           2.72          5.72          1.83          1.01
3  2018-11-01           5.26          7.63          3.00          1.37
4  2017-09-01           2.34          4.62          1.49          1.07

      X10/ million
0            1.45
1            1.35
2            1.34
3            1.97
4            1.23
```

# Suppression des doublons

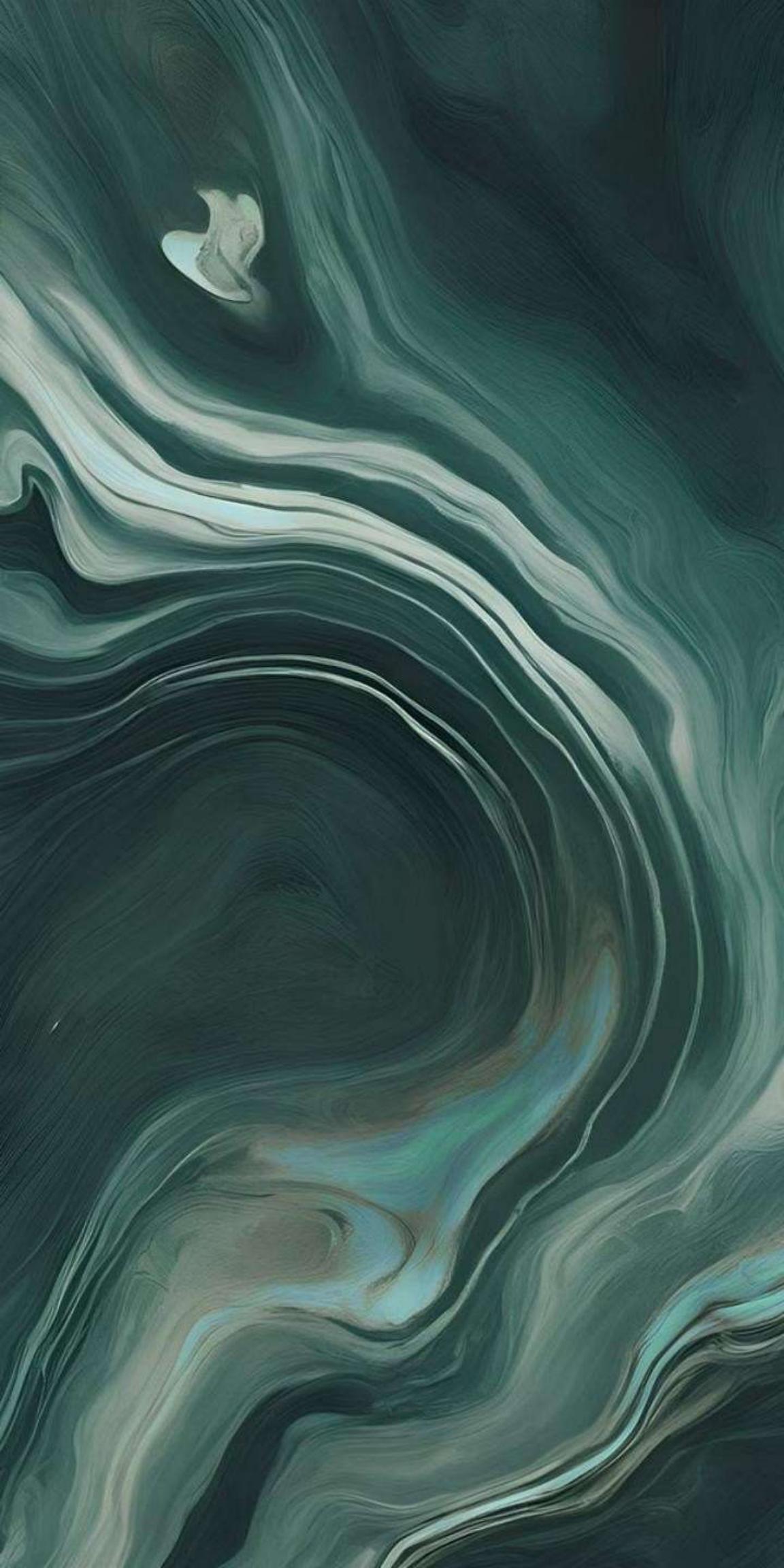
Le résultat du code:

Les résultats montrent que les doublons ont été correctement gérés, aucun doublon n'ayant été détecté dans les datasets des pantalons, robes et pulls.

```
Nombre de doublons supprimés dans Pantalons : 0
Nombre de doublons supprimés dans Robes : 0
Nombre de doublons supprimés dans Pulls : 0
Aperçu des données nettoyées (Pantalons):
    Date  S/ billion Yuan  X4/ billion  X8/ 10\nmillion  X9/ 100 thousand \
0  2019-04-01            3.12          5.45           1.95          1.04
1  2017-06-01            2.78          6.62           1.92          1.18
2  2018-07-01            2.72          5.72           1.83          1.01
3  2018-11-01            5.26          7.63           3.00          1.37
4  2017-09-01            2.34          4.62           1.49          1.07

    X10/ million
0            1.45
1            1.35
2            1.34
3            1.97
4            1.23
```

## 4—Gestion des valeurs manquantes



# Gestion des valeurs manquantes

Après avoir supprimé les doublons pour garantir l'unicité des enregistrements, l'étape suivante consiste à gérer les éventuelles valeurs manquantes.

L'imputation par la moyenne est choisie comme méthode, car elle permet de remplacer les valeurs manquantes par une estimation simple et cohérente, tout en préservant la distribution statistique des données numériques. Cette approche évite la perte d'informations due à une suppression des lignes incomplètes et garantit que les données restent exploitables pour les analyses ou l'entraînement des modèles

```
# Gestion des valeurs manquantes : Imputation par la moyenne

def impute_missing_values(df, name):
    for column in df.select_dtypes(include=["number"]).columns:
        if df[column].isnull().sum() > 0:
            # Remplacer les valeurs manquantes par la moyenne
            mean_value = df[column].mean()
            df[column].fillna(mean_value, inplace=True)
    return df

# Vérifier les données avant imputation
print("Aperçu des données avant imputation (Pantalons):")
print(df_pants_cleaned.head())
# Appliquer l'imputation sur chaque dataset nettoyé
df_pants_imputed = impute_missing_values(df_pants_cleaned, "Pantalons")
df_dress_imputed = impute_missing_values(df_dress_cleaned, "Robes")
df_sweater_imputed = impute_missing_values(df_sweater_cleaned, "Pulls")

# Vérifier les données après imputation
print("Aperçu des données après imputation (Pantalons):")
print(df_pants_imputed.head())
```

# Gestion des valeurs manquantes

Le résultat du code:

Aperçu des données avant imputation (Pantalons):						
	Date	S/ billion Yuan	X4/ billion	X8/ 10\nmillion	X9/ 100 thousand	\
0	2019-04-01	3.12	5.45	1.95	1.04	
1	2017-06-01	2.78	6.62	1.92	1.18	
2	2018-07-01	2.72	5.72	1.83	1.01	
3	2018-11-01	5.26	7.63	3.00	1.37	
4	2017-09-01	2.34	4.62	1.49	1.07	

Aperçu des données après imputation (Pantalons):						
	Date	S/ billion Yuan	X4/ billion	X8/ 10\nmillion	X9/ 100 thousand	\
0	2019-04-01	3.12	5.45	1.95	1.04	
1	2017-06-01	2.78	6.62	1.92	1.18	
2	2018-07-01	2.72	5.72	1.83	1.01	
3	2018-11-01	5.26	7.63	3.00	1.37	
4	2017-09-01	2.34	4.62	1.49	1.07	

Aperçu des données après imputation (Pantalons):						
	Date	S/ billion Yuan	X4/ billion	X8/ 10\nmillion	X9/ 100 thousand	\
0	2019-04-01	3.12	5.45	1.95	1.04	
1	2017-06-01	2.78	6.62	1.92	1.18	
2	2018-07-01	2.72	5.72	1.83	1.01	
3	2018-11-01	5.26	7.63	3.00	1.37	
4	2017-09-01	2.34	4.62	1.49	1.07	

Aperçu des données après imputation (Pantalons):						
	Date	S/ billion Yuan	X4/ billion	X8/ 10\nmillion	X9/ 100 thousand	\
0	2019-04-01	3.12	5.45	1.95	1.04	
1	2017-06-01	2.78	6.62	1.92	1.18	
2	2018-07-01	2.72	5.72	1.83	1.01	
3	2018-11-01	5.26	7.63	3.00	1.37	
4	2017-09-01	2.34	4.62	1.49	1.07	

Aperçu des données après imputation (Pantalons):						
	Date	S/ billion Yuan	X4/ billion	X8/ 10\nmillion	X9/ 100 thousand	\
0	2019-04-01	3.12	5.45	1.95	1.04	
1	2017-06-01	2.78	6.62	1.92	1.18	
2	2018-07-01	2.72	5.72	1.83	1.01	
3	2018-11-01	5.26	7.63	3.00	1.37	
4	2017-09-01	2.34	4.62	1.49	1.07	

Aperçu des données après imputation (Pantalons):						
	Date	S/ billion Yuan	X4/ billion	X8/ 10\nmillion	X9/ 100 thousand	\
0	2019-04-01	3.12	5.45	1.95	1.04	
1	2017-06-01	2.78	6.62	1.92	1.18	
2	2018-07-01	2.72	5.72	1.83	1.01	
3	2018-11-01	5.26	7.63	3.00	1.37	
4	2017-09-01	2.34	4.62	1.49	1.07	

Aperçu des données après imputation (Pantalons):						
	Date	S/ billion Yuan	X4/ billion	X8/ 10\nmillion	X9/ 100 thousand	\
0	2019-04-01	3.12	5.45	1.95	1.04	
1	2017-06-01	2.78	6.62	1.92	1.18	
2	2018-07-01	2.72	5.72	1.83	1.01	
3	2018-11-01	5.26	7.63	3.00	1.37	
4	2017-09-01	2.34	4.62	1.49	1.07	

Aperçu des données après imputation (Pantalons):						
	Date	S/ billion Yuan	X4/ billion	X8/ 10\nmillion	X9/ 100 thousand	\
0	2019-04-01	3.12	5.45	1.95	1.04	
1	2017-06-01	2.78	6.62	1.92	1.18	
2	2018-07-01	2.72	5.72	1.83	1.01	
3	2018-11-01	5.26	7.63	3.00	1.37	
4	2017-09-01	2.34	4.62	1.49	1.07	

Étant donné que le dataset ne contenait aucune valeur manquante dans les colonnes numériques, l'application de l'imputation par la moyenne n'a eu aucun impact. Les données sont donc restées inchangées.

## 5—Détection des valeurs aberrantes



# Détection des valeurs aberrantes

Après la gestion des valeurs manquantes, l'étape suivante consiste à détecter et analyser les valeurs aberrantes présentes dans les données. Ces valeurs, souvent situées en dehors de la plage normale des observations, peuvent fausser les analyses statistiques ou les modèles prédictifs. Pour ce faire, nous combinons une visualisation avec des boîtes à moustaches pour une détection visuelle des anomalies et la méthode de l'Intervalle Interquartile (IQR) pour une identification quantitative. Cette approche nous permettra d'évaluer l'étendue des valeurs extrêmes et de décider des actions correctives nécessaires.

```
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import zscore
import numpy as np

# Fonction pour détecter les valeurs aberrantes avec des boîtes à moustaches et des scores Z
def detect_outliers(df, name):
    numeric_columns = df.select_dtypes(include=["number"]).columns

    # Visualisation avec des boîtes à moustaches
    plt.figure(figsize=(12, 6))
    sns.boxplot(data=df[numeric_columns])
    plt.title("Boîtes à moustaches pour " + name)
    plt.xticks(rotation=45)
    plt.show()

    # Méthode IQR
    Q1 = df[numeric_columns].quantile(0.25)
    Q3 = df[numeric_columns].quantile(0.75)
    IQR = Q3 - Q1
    outliers_iqr = ((df[numeric_columns] < (Q1 - 1.5 * IQR)) | (df[numeric_columns] > (Q3 + 1.5 * IQR))).any(axis=1)
    print("Nombre de valeurs aberrantes détectées avec la méthode IQR dans", name, ":", outliers_iqr.sum())

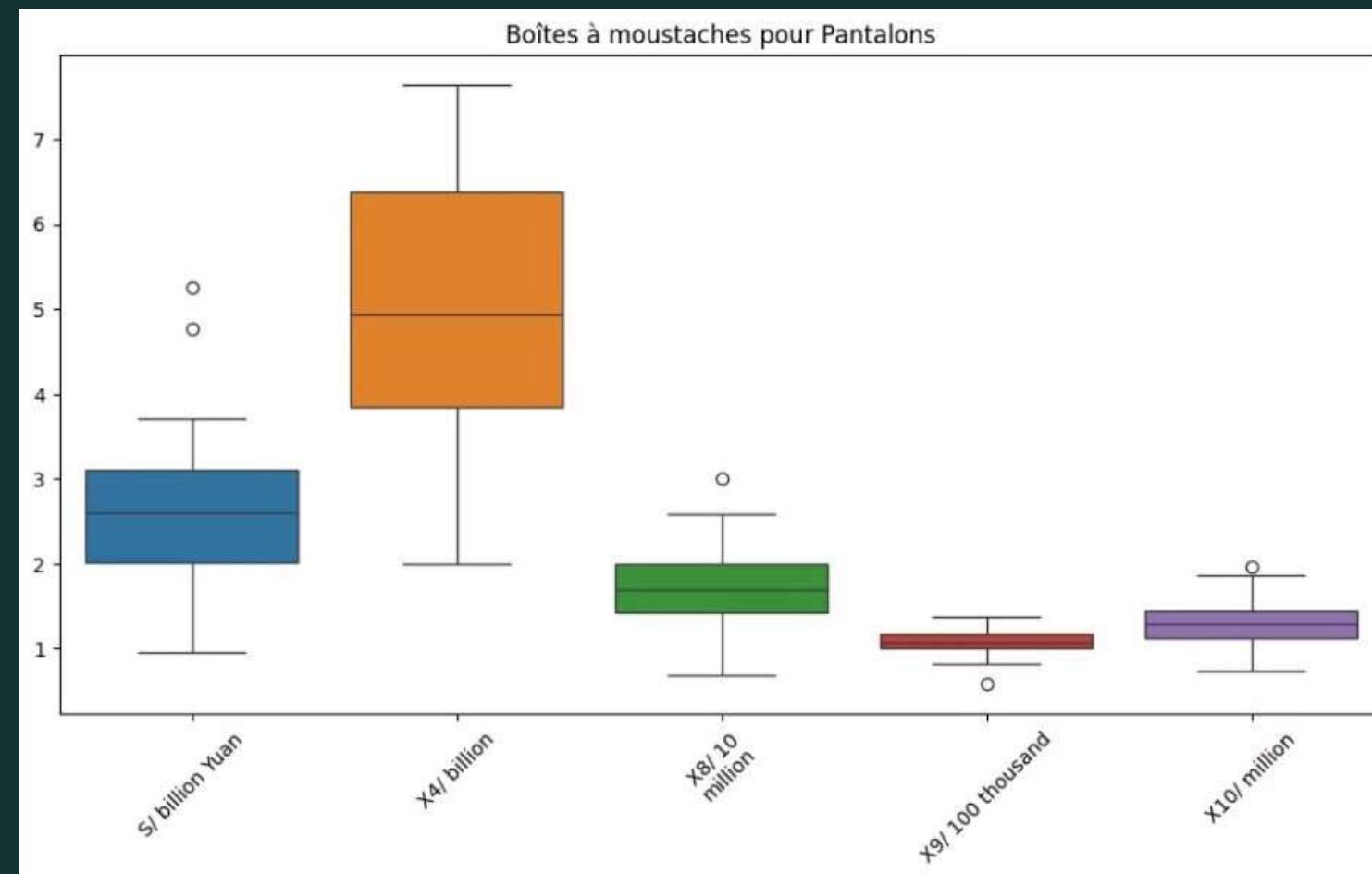
    # Appliquer la détection des valeurs aberrantes sur chaque dataset
detect_outliers(df_pants_imputed, "Pantalons")
detect_outliers(df_dress_imputed, "Robes")
detect_outliers(df_sweater_imputed, "Pulls")
```

# Détection des valeurs aberrantes

Le graphique des boîtes à moustaches met en évidence 5 valeurs aberrantes dans le dataset "Pantalons". Ces points isolés apparaissent comme suit :

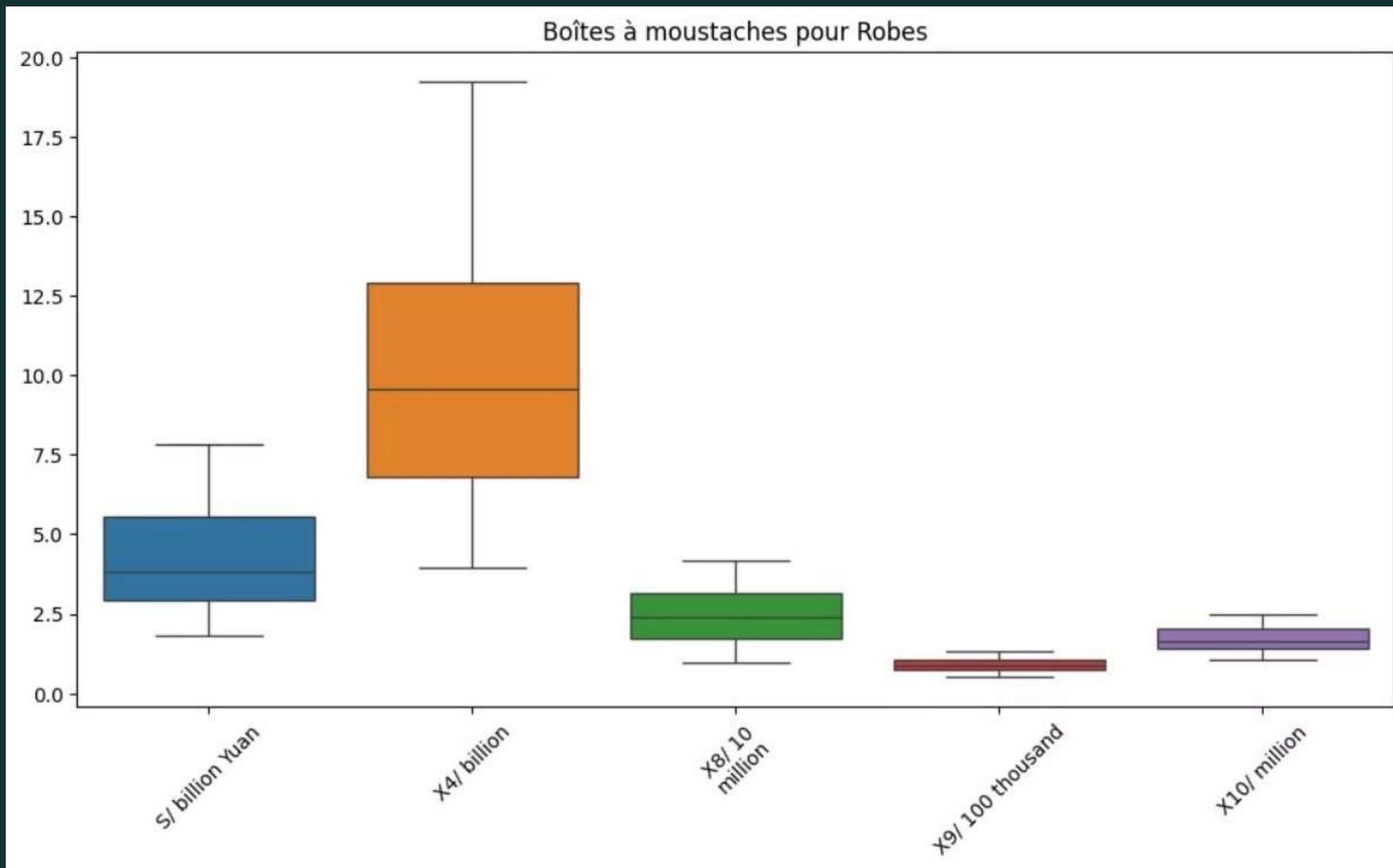
- Deux dans la colonne "S/ billion Yuan".
- Une dans "X8/10 million".
- Une dans "X9/100 thousand".
- Une dans "X10/ million".

En revanche, la méthode IQR a détecté uniquement 3 valeurs aberrantes. Cette différence peut s'expliquer par le fait que la méthode IQR utilise un seuil statistique rigoureux basé sur les clôtures  $F1 = (Q1 - 1.5 * IQR)$  et  $F3 = (Q3 + 1.5 * IQR)$ , tandis que la visualisation avec les boîtes à moustaches peut inclure des valeurs qui s'approchent de ces seuils mais ne les dépassent pas strictement. Cela souligne l'importance de combiner les deux approches pour une analyse complète des valeurs aberrantes.



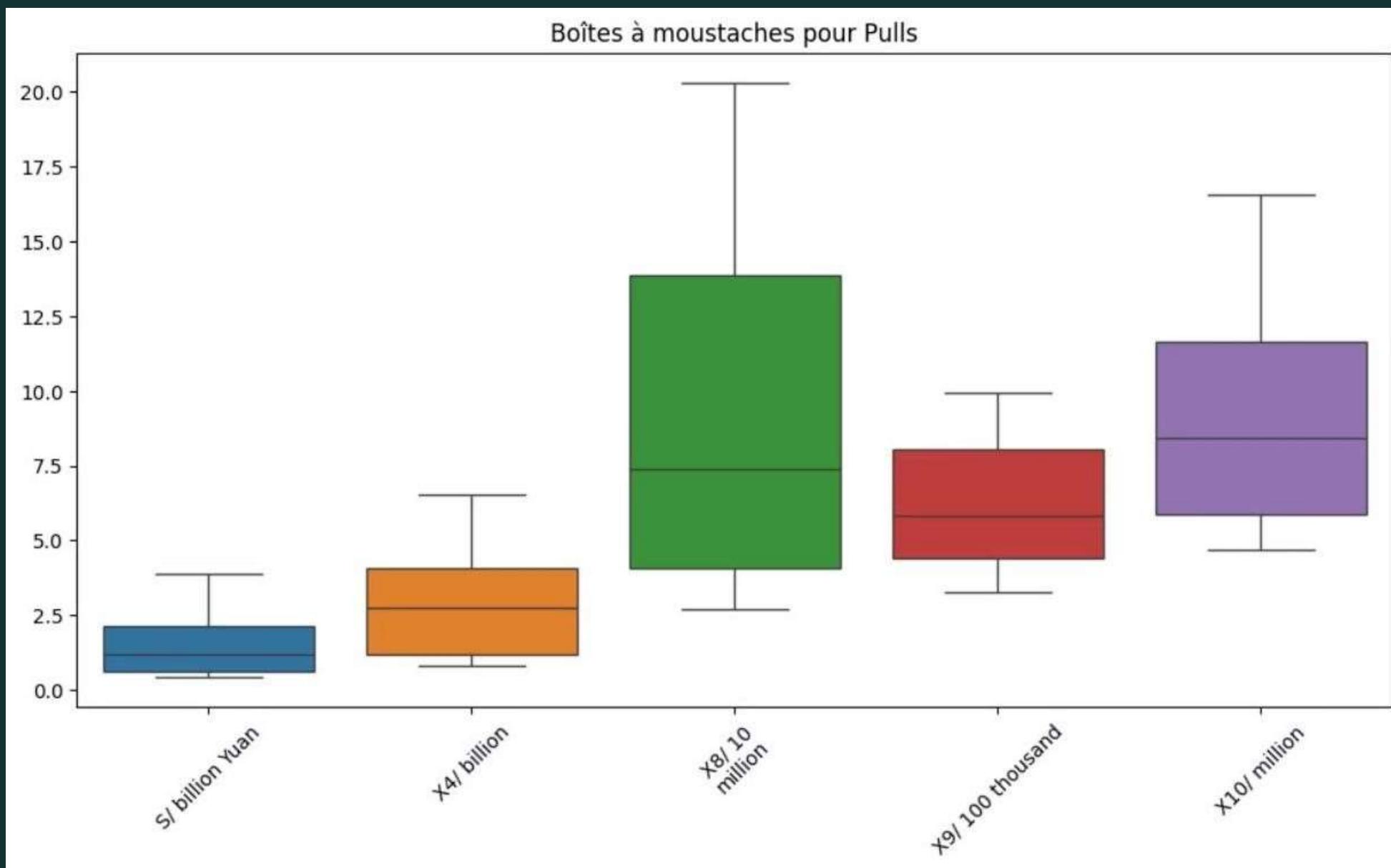
Nombre de valeurs aberrantes détectées avec la méthode IQR dans Pantalons : 3

# Détection des valeurs aberrantes



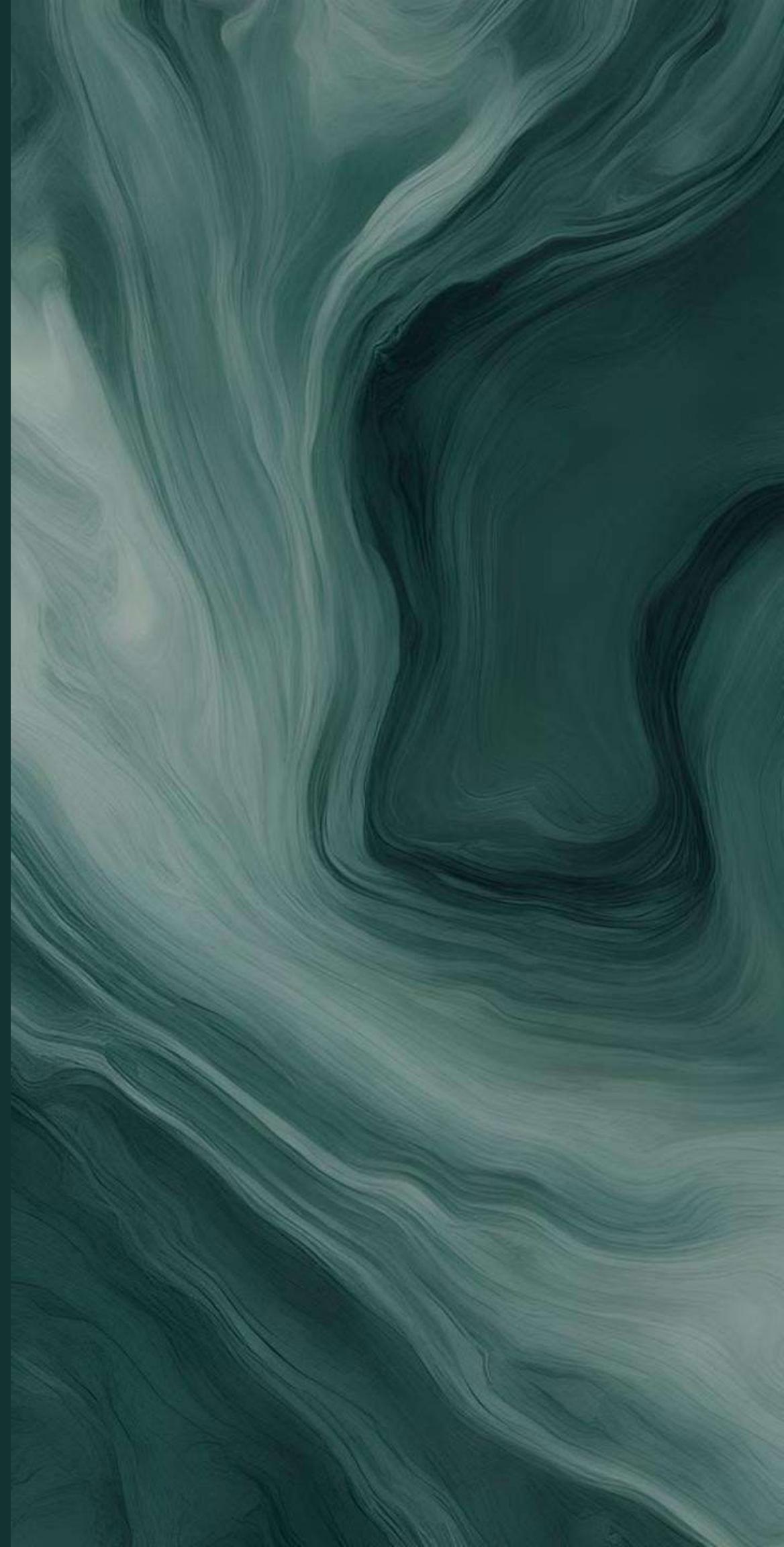
Nombre de valeurs aberrantes détectées avec la méthode IQR dans Robes : 0

# Détection des valeurs aberrantes



Nombre de valeurs aberrantes détectées avec la méthode IQR dans Pulls : 0

## 6—Mise à l'échelle des caractéristiques



# Mise à l'échelle des caractéristiques

Après avoir détecté et traité les valeurs aberrantes, l'étape suivante consiste à standardiser les données.

La standardisation des données transforme chaque variable pour avoir une moyenne de 0 et un écart-type de 1, ce qui homogénéise l'échelle des variables. Cela permet d'éviter qu'une variable avec une grande plage de valeurs n'influence indûment les analyses ou les modèles. Elle est essentielle pour améliorer la performance.

```
from sklearn.preprocessing import StandardScaler

def standardize_data(df, columns_to_standardize):

    # Initialisation du StandardScaler
    scaler = StandardScaler()
    # Appliquer la standardisation aux colonnes spécifiées
    df[columns_to_standardize] = scaler.fit_transform(df[columns_to_standardize])

    return df

# Nettoyer les noms de colonnes en remplaçant les caractères indésirables
df_pants_imputed.columns = df_pants_imputed.columns.str.replace("\n", " ", regex=False)
df_dress_imputed.columns = df_dress_imputed.columns.str.replace("\n", " ", regex=False)
df_sweater_imputed.columns = df_sweater_imputed.columns.str.replace("\n", " ", regex=False)

# Vérification des noms de colonnes après nettoyage
print(df_pants_imputed.columns)

# Liste des colonnes numériques à standardiser (mises à jour avec les bons noms de colonnes)
columns_to_standardize = ["S/ billion Yuan", "X4/ billion", "X8/ 10 million", "X9/ 100 thousand", "X10/ million"]

# Appliquer la standardisation sur les datasets imputés
df_pants_standardized = standardize_data(df_pants_imputed, columns_to_standardize)
df_dress_standardized = standardize_data(df_dress_imputed, columns_to_standardize)
df_sweater_standardized = standardize_data(df_sweater_imputed, columns_to_standardize)
```

# Mise à l'échelle des caractéristiques

```
# Vérification des résultats
print("Données standardisées pour Pantalons:")
print(df_pants_standardized.head(2))

print("\nDonnées standardisées pour Robes:")
print(df_dress_standardized.head(2))

print("\nDonnées standardisées pour Pulls:")
print(df_sweater_standardized.head(2))
```

# Mise à l'échelle des caractéristiques

```
Données standardisées pour Pantalons:  
Date S/ billion Yuan X4/ billion X8/ 10 million X9/ 100 thousand \  
0 2019-04-01 0.437344 0.292520 0.413253 -0.244695  
1 2017-06-01 0.092630 1.084261 0.358485 0.565694  
  
X10/ million  
0 0.510462  
1 0.151671  
  
Données standardisées pour Robes:  
Date S/ billion Yuan X4/ billion X8/ 10 million X9/ 100 thousand \  
0 2017-10-01 -0.991589 -0.726418 -0.910284 -0.792817  
1 2017-06-01 1.465207 1.910203 1.491649 1.777579  
  
X10/ million  
0 -1.003872  
1 1.417749  
  
Données standardisées pour Pulls:  
Date S/ billion Yuan X4/ billion X8/ 10 million X9/ 100 thousand \  
0 2019-04-01 -0.357753 -0.379520 -0.420737 -0.392319  
1 2017-06-01 -0.954009 -0.990314 -1.020223 -0.991286  
  
X10/ million  
0 -0.237613  
1 -1.017668
```

## 7—Relation entre les caractéristiques et les ventes



# Relation entre les caractéristiques et les ventes

L'étape suivante consiste à visualiser la courbe de relation entre les caractéristiques des données et les ventes, afin d'examiner visuellement les tendances et les corrélations possibles. Cette étape est cruciale pour mieux comprendre comment chaque caractéristique interagit avec les ventes après la standardisation des données.

```
\def plot_all_features_vs_sales_with_lines(datasets, sales_column, titles, output_filename):

    num_datasets = len(datasets)
    max_features = max(len(df.select_dtypes(include=["number"]).columns) - 1 for df in datasets)
    plt.figure(figsize=(20, 5 * num_datasets))

    for dataset_idx, (df, title) in enumerate(zip(datasets, titles), start=1):
        numeric_columns = df.select_dtypes(include=["number"]).columns
        numeric_columns = [col for col in numeric_columns if col != sales_column]

        for feature_idx, column in enumerate(numeric_columns, start=1):
            plt.subplot(num_datasets, max_features, (dataset_idx - 1) * max_features + feature_idx)

            # Tracer les points
            sns.scatterplot(data=df, x=column, y=sales_column, color="blue", label="Points")

            # Tracer les lignes reliant les points
            sns.lineplot(data=df.sort_values(by=column), x=column, y=sales_column, color="red", label="Lignes")

            plt.title(f"{title}: {column} vs {sales_column}")
            plt.xlabel(column)
            plt.ylabel(sales_column)
            plt.legend()

            plt.tight_layout()
            plt.savefig(output_filename, dpi=300)
            print(f"Graphiques sauvegardés sous {output_filename}")
```

# Relation entre les caractéristiques et les ventes

```
# Définir le nom de la colonne représentant les ventes
sales_column = "S/ billion Yuan"

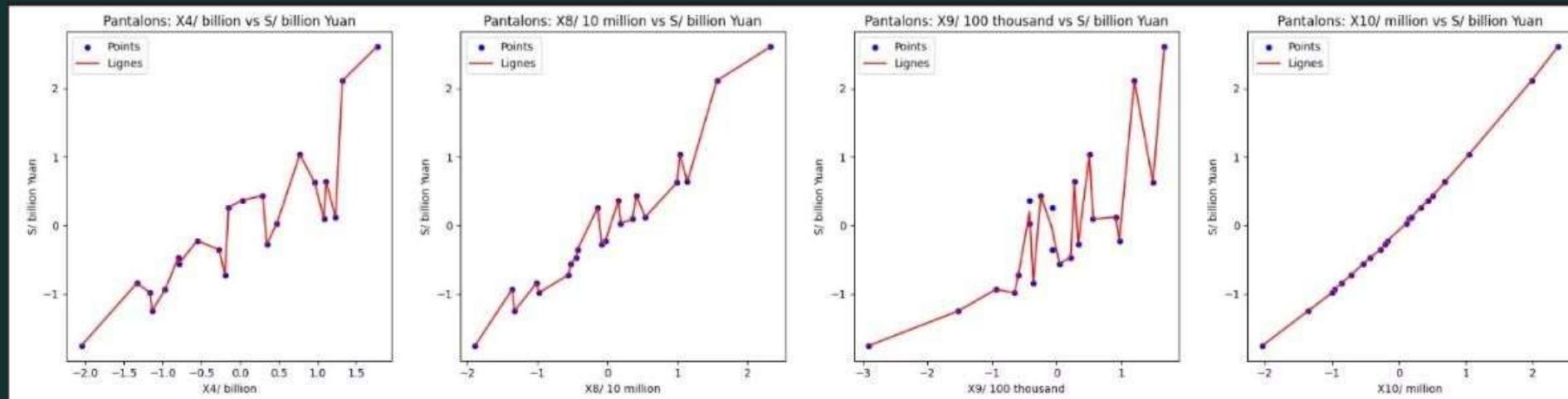
# Appliquer aux datasets
datasets = [df_pants_standardized, df_dress_standardized, df_sweater_standardized]
titles = ["Pantalons", "Robes", "Pulls"]
output_filename = "All_Features_vs_Sales_With_Lines.png"

plot_all_features_vs_sales_with_lines(datasets, sales_column, titles, output_filename)
```

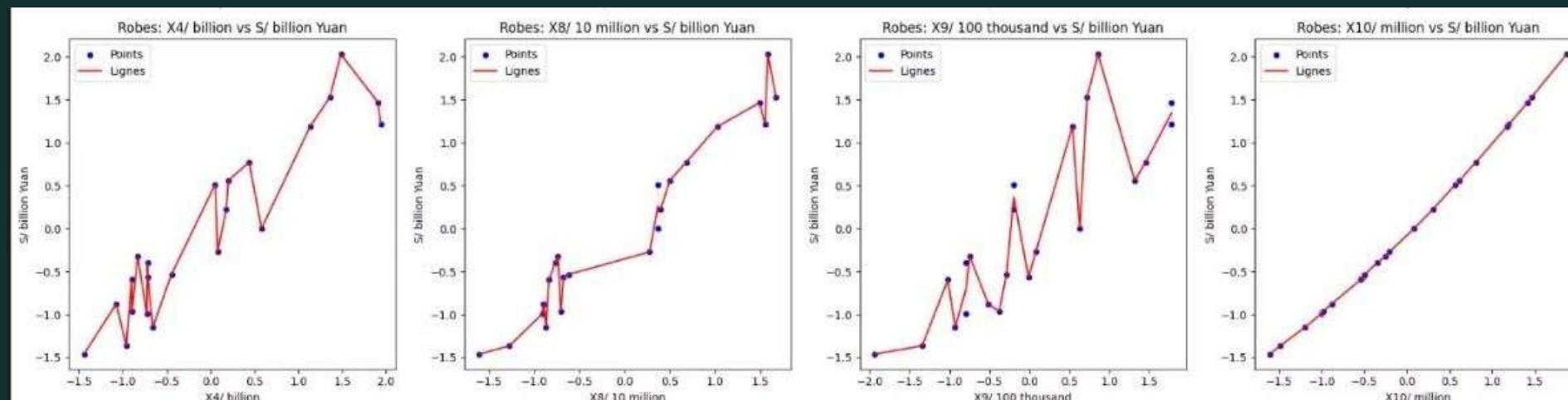
# Relation entre les caractéristiques et les ventes

## Courbes:

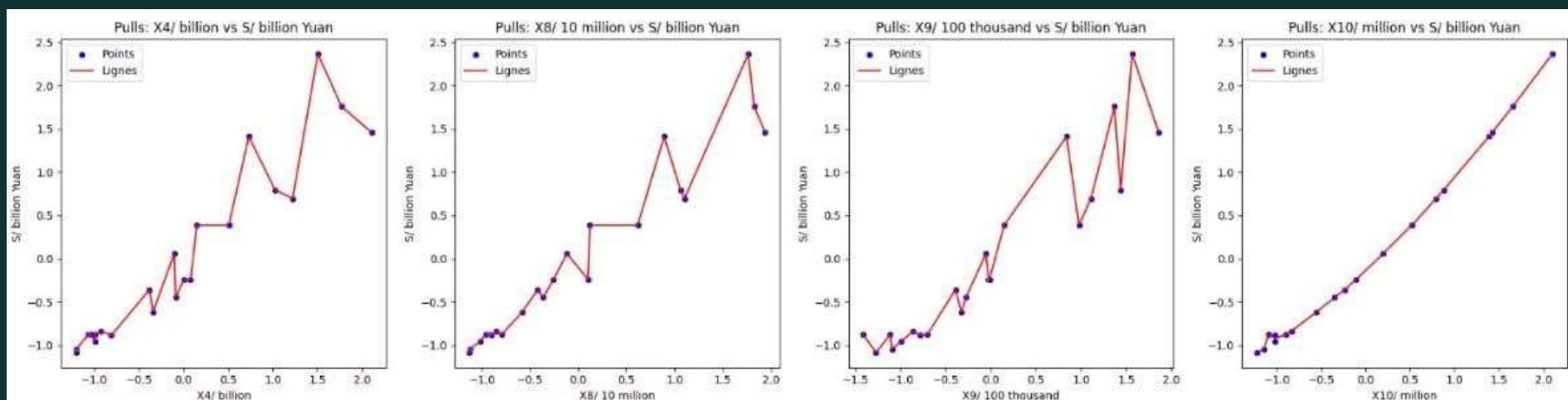
les courbes de relation entre les caractéristiques des données et les ventes pour les **Pantalons**:



les courbes de relation entre les caractéristiques des données et les ventes pour les **Robes**:



les courbes de relation entre les caractéristiques des données et les ventes pour les **Pulls**:



# Relation entre les caractéristiques et les ventes

## Interprétations:

Ces courbes montrent la relation entre diverses caractéristiques (X4, X8, X9, X10) et les ventes (S) pour trois catégories de produits : pantalons, robes et pulls.

Voici une interprétation générale pour chaque cas :

### 1 Pantalons :

- **X4 (milliards)** : Une tendance générale positive, avec quelques oscillations. Cela indique que cette caractéristique a une influence globale croissante sur les ventes, mais les fluctuations peuvent signifier une relation non linéaire ou des facteurs externes.
- **X8 (millions)** : Une relation claire et positive. Plus cette caractéristique augmente, plus les ventes augmentent.
- **X9 (milliers)** : Une relation moins stable avec des pics et des baisses significatifs, indiquant une sensibilité accrue des ventes à cette caractéristique.
- **X10 (millions)** : Une relation quasi-linéaire, montrant une corrélation directe forte entre cette caractéristique et les ventes.

### 2 Robes :

- **X4 (milliards)** : Similaire aux pantalons, une relation positive mais légèrement plus fluctuante, indiquant une variabilité plus marquée dans l'impact de cette caractéristique.
- **X8 (millions)** : Relation positive et constante. Cela suggère que cette caractéristique influence les ventes de manière prévisible.
- **X9 (milliers)** : Fluctuations importantes, mais avec une tendance croissante. Cela peut indiquer une interaction complexe entre cette caractéristique et les ventes.
- **X10 (millions)** : Relation linéaire forte et positive, montrant un effet direct et constant sur les ventes.

### 3 Pulls :

- **X4 (milliards)** : Une tendance globalement croissante, avec des variations modérées. Cette caractéristique a un effet positif mais non uniforme sur les ventes.
- **X8 (millions)** : Relation linéaire claire et croissante, similaire aux robes.
- **X9 (milliers)** : Des oscillations significatives, mais une tendance générale positive. Cela peut refléter une dépendance plus complexe.
- **X10 (millions)** : Une forte relation linéaire directe, similaire aux pantalons et robes.

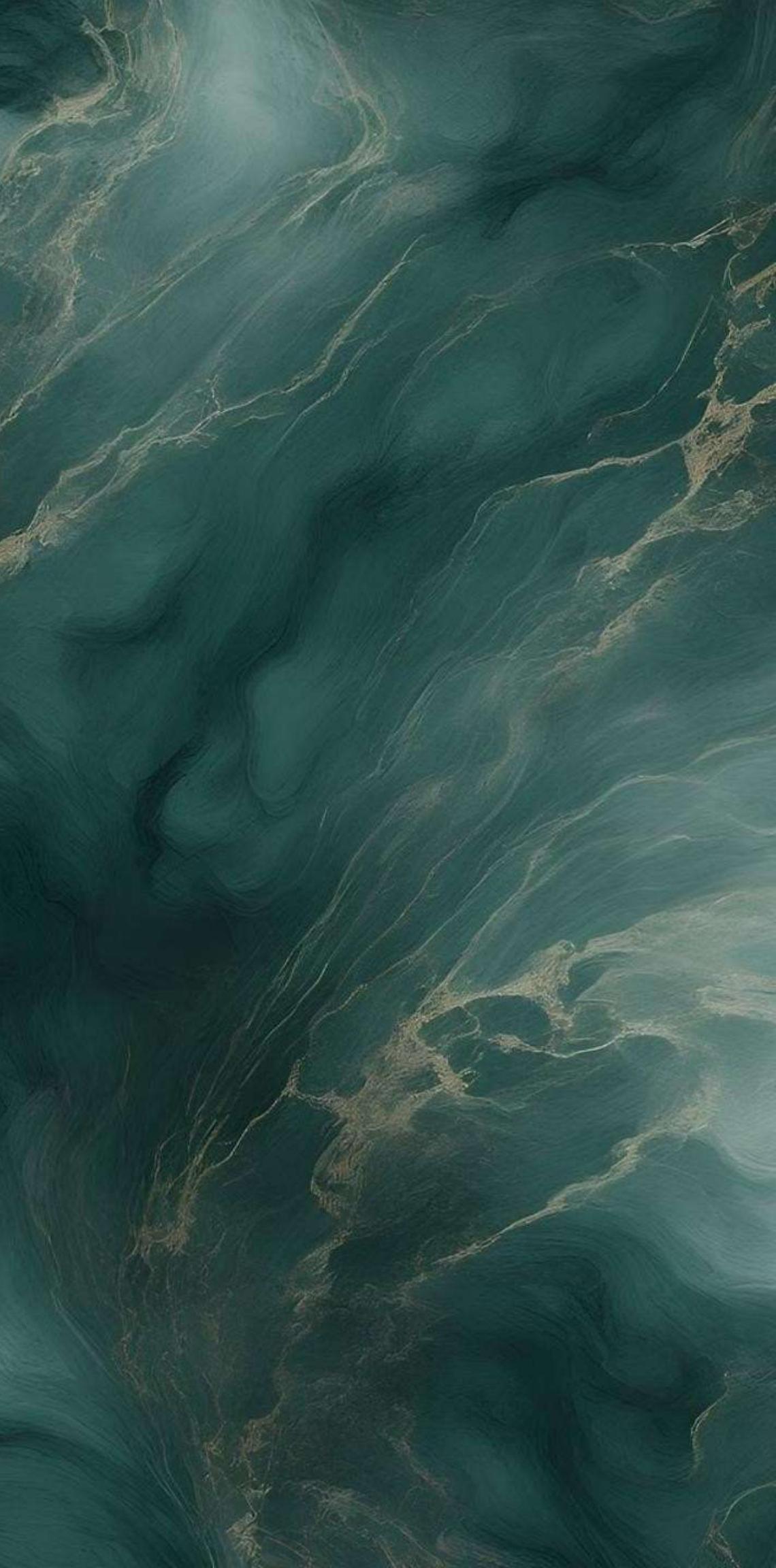
# Relation entre les caractéristiques et les ventes

## Conclusions:

1. Les caractéristiques **X8** et **X10** ont des relations claires et linéaires avec les ventes pour toutes les catégories, ce qui indique qu'elles sont des indicateurs fiables des performances de vente.
2. Les caractéristiques **X4** et **X9** montrent plus de variations, indiquant qu'elles peuvent être influencées par d'autres facteurs ou qu'elles nécessitent une analyse approfondie pour modéliser correctement leur impact.
3. Les oscillations visibles dans certaines courbes pourraient refléter des effets de seuils ou des variations saisonnières/non linéaires dans la demande.

Ces observations peuvent être utilisées pour affiner les stratégies marketing ou améliorer les modèles prédictifs des ventes pour chaque catégorie de produit.

## 8—La corrélation entre les différentes variables et les ventes



# La corrélation entre les différentes variables et les ventes

Après avoir visualisé la courbe de relation entre les caractéristiques et les ventes, l'étape suivante consiste à étudier la corrélation entre les différentes variables et les ventes. Cela permet d'identifier les caractéristiques qui ont le plus d'impact sur les ventes de vêtements, en se concentrant sur celles qui montrent une relation forte et significative. Pour ce faire, une carte de corrélation (heatmap) est utilisée, qui affiche visuellement les corrélations entre toutes les variables et les ventes.

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

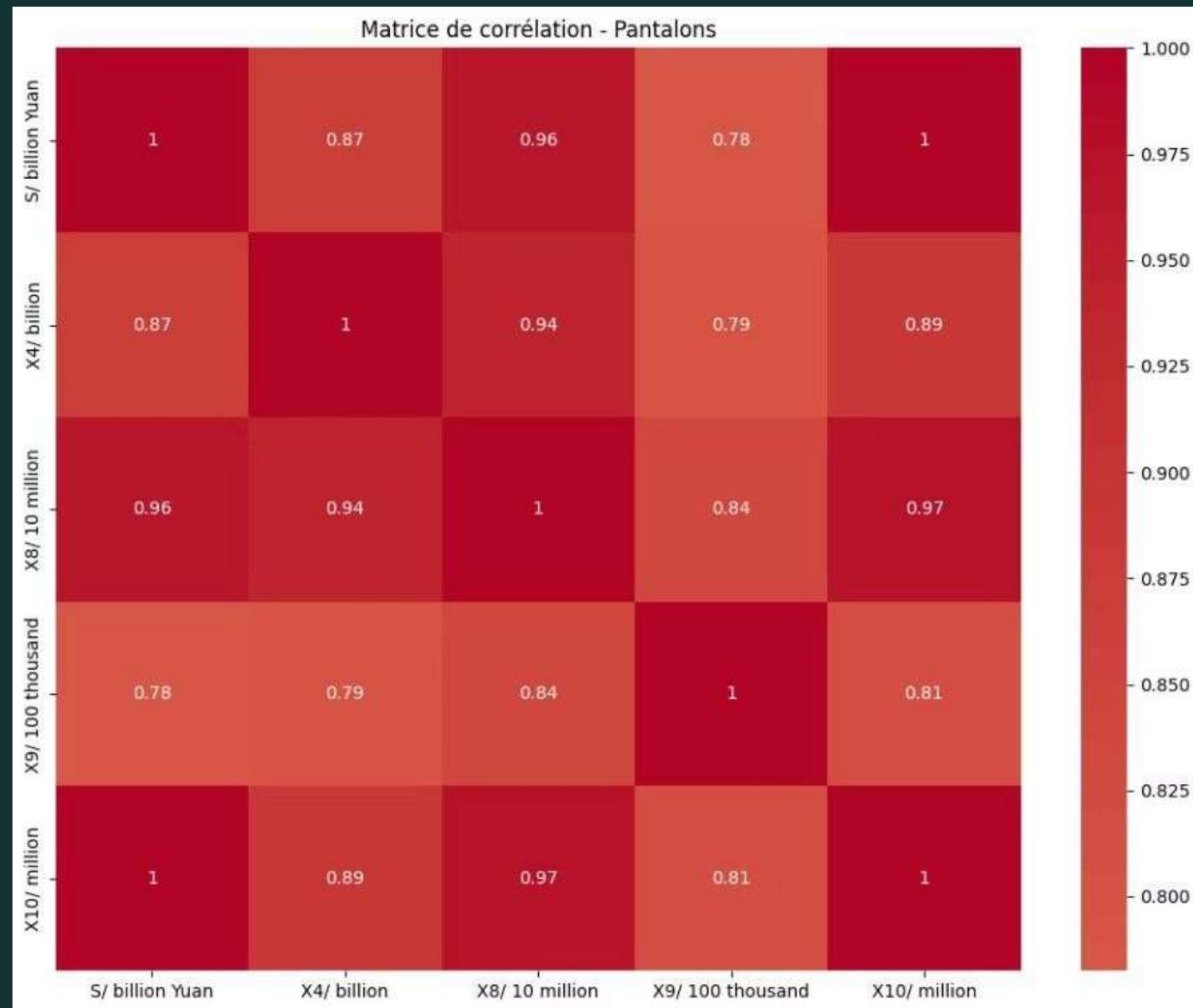
def plot_correlation_heatmap(df, title):
    # Sélectionner uniquement les colonnes numériques
    numeric_cols = df.select_dtypes(include=['float64', 'int64']).columns
    correlation_matrix = df[numeric_cols].corr()

    plt.figure(figsize=(10, 8))
    sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', center=0)
    plt.title(f'Matrice de corrélation - {title}')
    plt.tight_layout()
    plt.show()

# Créer les heatmaps pour chaque type de vêtement
plot_correlation_heatmap(df_pants_standardized, 'Pantalons')
plot_correlation_heatmap(df_dress_standardized, 'Robes')
plot_correlation_heatmap(df_sweater_standardized, 'Pulls')
```

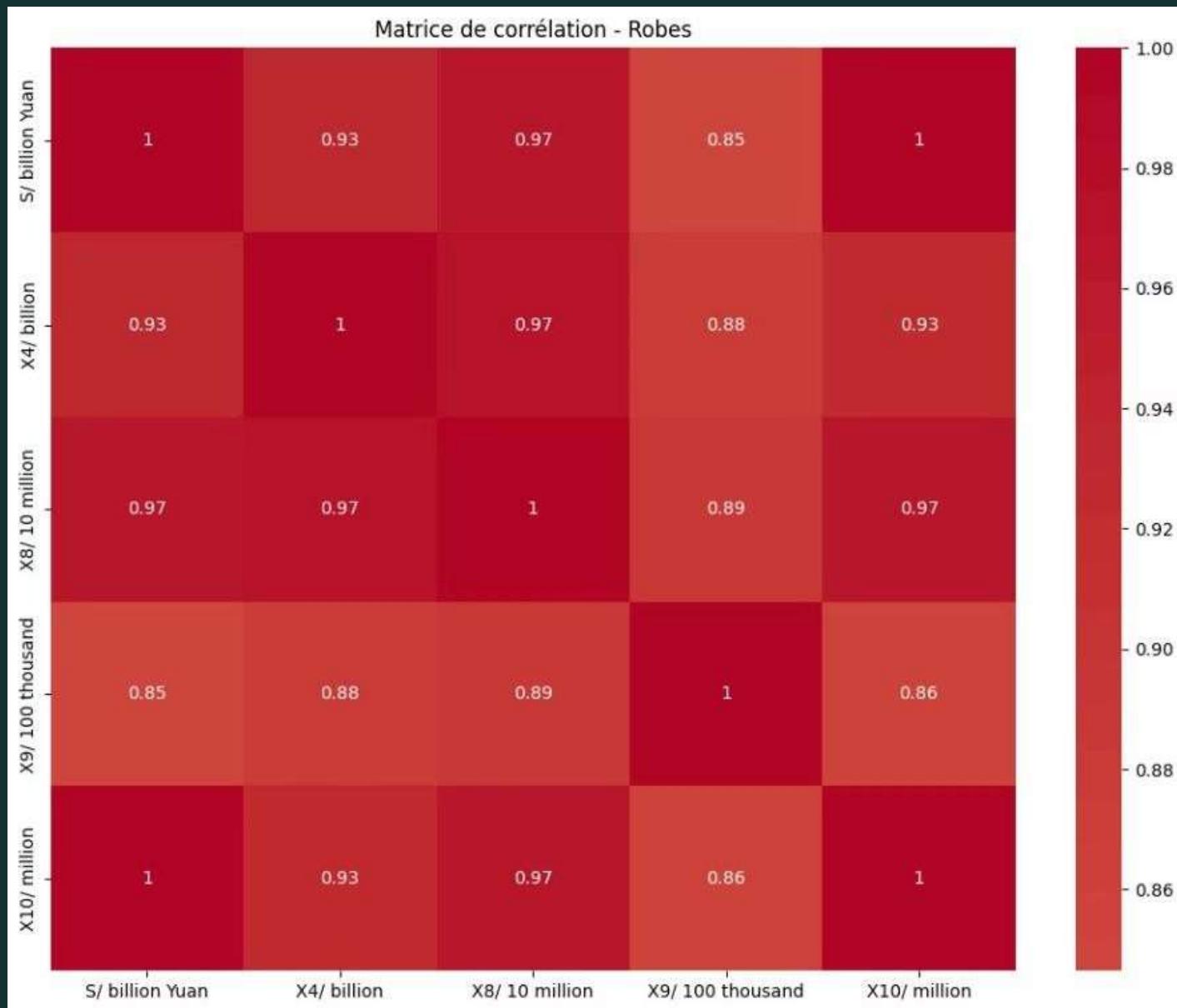
# La corrélation entre les différentes variables et les ventes

## Carte de corrélation pour Pantalons:



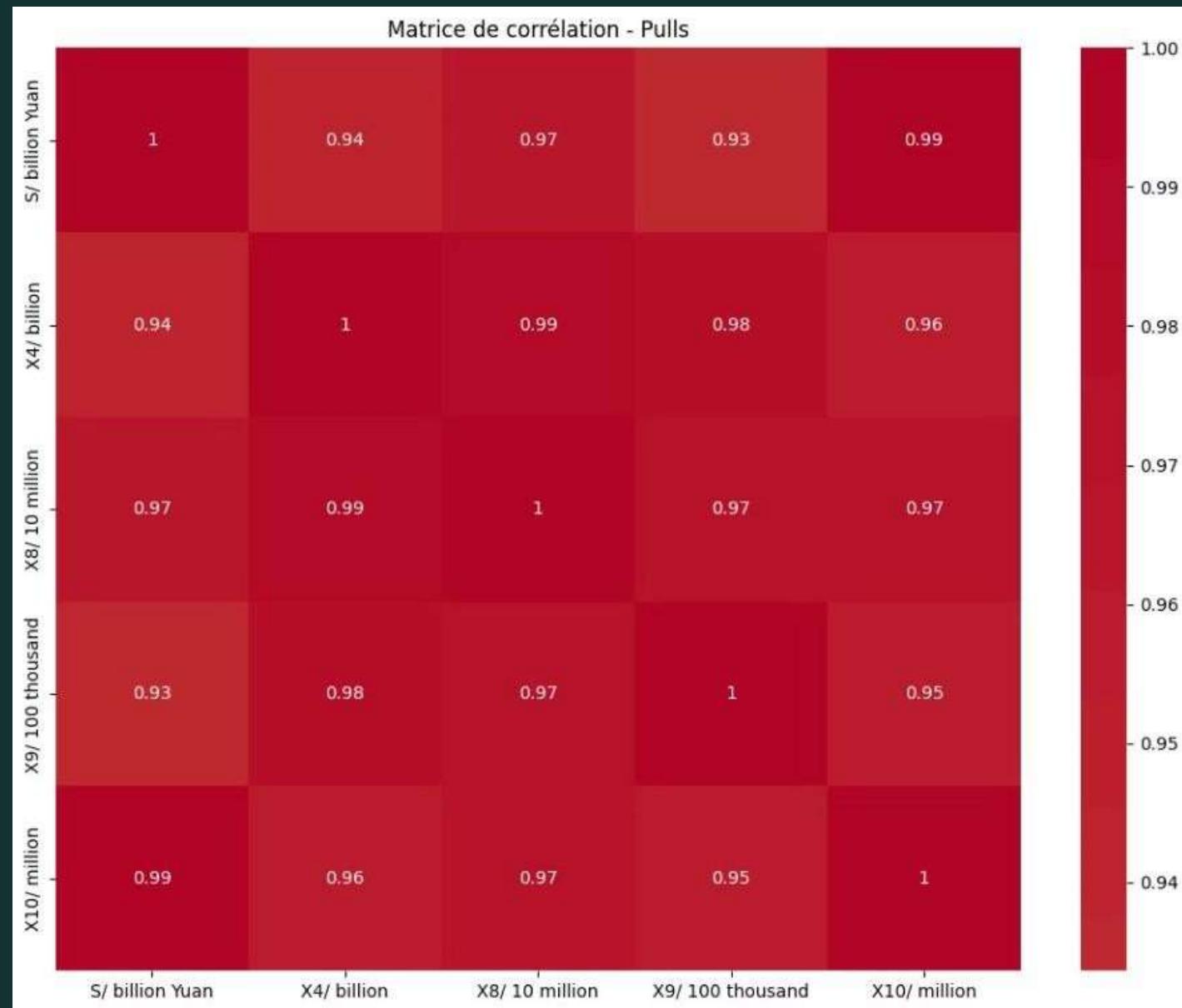
# La corrélation entre les différentes variables et les ventes

## Carte de corrélation pour Robes:



# La corrélation entre les différentes variables et les ventes

## Carte de corrélation pour Pulls:



# La corrélation entre les différentes variables et les ventes

## Interprétations:

Les cartes thermiques permettent d'évaluer la relation entre différentes variables et les ventes (S/ billion Yuan) pour trois types de produits : pantalons, robes et pulls. Voici une interprétation des corrélations pour chaque cas :

### 1 Pantalons :

- **Variables avec une forte corrélation positive :**
- X8 (10 million) : Corrélation de 0.96 avec les ventes, ce qui indique que l'augmentation de cette variable est fortement liée à une hausse des ventes.
- X10 (million) : Corrélation parfaite (1.0), ce qui signifie une relation linéaire directe avec les ventes.
- **Variables modérément corrélées :**
- X4 (billion) : Corrélation de 0.87.
- X9 (100 thousand) : Corrélation plus faible à 0.78.
- **Interprétation générale :** Les ventes de pantalons dépendent fortement des variables mesurées en millions et en milliards, ce qui peut refléter leur contribution significative au chiffre d'affaires.

### 2 Robes :

- **Variables avec une très forte corrélation positive :**
- X8 (10 million) et X10 (million) : Corrélations de 0.97 et 1.0 respectivement, indiquant une forte dépendance des ventes à ces variables.
- X4 (billion) : Corrélation élevée à 0.93.
- **Variables avec une corrélation modérée :**
- X9 (100 thousand) : Corrélation de 0.85, légèrement plus faible mais toujours positive.
- **Interprétation générale :** Les ventes de robes montrent une forte relation avec les mesures en millions et en milliards, tout en ayant une dépendance légèrement plus faible avec les unités plus petites.

### 3 Pulls :

- **Variables avec une forte corrélation positive :**
- X4, X8 et X10 : Corrélations respectives de 0.94, 0.97 et 0.99 avec les ventes. Cela suggère une relation étroite et linéaire.
- **Variables avec une corrélation légèrement plus faible :**
- X9 (100 thousand) : Corrélation de 0.93.
- **Interprétation générale :** Les ventes de pulls montrent une cohérence marquée dans leur relation avec les différentes variables, particulièrement celles mesurées en millions.

# La corrélation entre les différentes variables et les ventes

## Conclusions:

Pour les trois produits, les ventes (S/ billion Yuan) sont fortement influencées par les variables représentant des chiffres d'affaires élevés (X8 et X10). Cela indique que les grands nombres d'achats supplémentaires ou les grosses transactions contribuent principalement aux ventes globales. Les corrélations légèrement plus faibles pour X9 (100 thousand) suggèrent que les petits montants ou segments jouent un rôle moindre.

## 9—Test d'hypothèse



# Test de "Student"

On veut analyser les ventes de trois catégories de produits : pantalons, robes et pulls. Pour ceci on a opté pour le test de Student afin de comparer les moyennes des ventes entre chaque paire de catégories de produits. Le but est de déterminer s'il existe des différences significatives dans les ventes des produits.

```
pants_sales = df_pants_cleaned['$/ billion Yuan']
dress_sales = df_dress_cleaned['$/ billion Yuan']
sweater_sales = df_sweater_cleaned['$/ billion Yuan']

# Test de Student entre les pantalons et les robes
t_stat_1, p_value_1 = stats.ttest_ind(pants_sales, dress_sales, nan_policy='omit')
print(f"Test de Student (Pantalons vs Robes) :")
print(f"t-statistique : {t_stat_1}, p-value : {p_value_1}")

# Test de Student entre les pantalons et les pulls
t_stat_2, p_value_2 = stats.ttest_ind(pants_sales, sweater_sales, nan_policy='omit')
print(f"\nTest de Student (Pantalons vs Pulls) :")
print(f"t-statistique : {t_stat_2}, p-value : {p_value_2}")

# Test de Student entre les robes et les pulls
t_stat_3, p_value_3 = stats.ttest_ind(dress_sales, sweater_sales, nan_policy='omit')
print(f"\nTest de Student (Robes vs Pulls) :")
print(f"t-statistique : {t_stat_3}, p-value : {p_value_3:.10f}")
```

# Tests d'hypothèses

Le test de Student (ou test t) permet de tester l'hypothèse nulle selon laquelle les moyennes de deux échantillons sont égales. L'hypothèse alternative stipule que les moyennes sont différentes. Pour chaque paire de produits, nous avons formulé les hypothèses suivantes :

1

## Pantalons vs Robes

**(H0)** : Les moyennes des ventes de pantalons et de robes sont égales.

**(H1)** : Les moyennes des ventes de pantalons et de robes sont différentes

2

## Pantalons vs Pulls

**(H0)** : Les moyennes des ventes de pantalons et de pulls sont égales.

**(H1)** : Les moyennes des ventes de pantalons et de pulls sont différentes.

3

## Robes vs Pulls

**(H0)** : Les moyennes des ventes de robes et de pulls sont égales.

**(H1)** : Les moyennes des ventes de robes et de pulls sont différentes

# p - value

La p-value est utilisée pour déterminer si nous rejetons l'hypothèse nulle ou non.

On choisit un **seuil de 0,05** qui signifie que nous acceptons un **risque de 5 %** de rejeter à tort l'hypothèse nulle.

Si  $p\text{-value} < 0,05$  on rejette  $H_0$ , ce qui indique que la différence observée est statistiquement significative.

Si  $p\text{-value} \geq 0,05$ , on ne rejette pas  $H_0$ , ce qui signifie que les preuves sont insuffisantes pour conclure à une différence significative.

```
Test de Student (Pantalons vs Robes) :  
t-statistique : -3.784375716250801, p-value : 0.0004828278438422783  
  
Test de Student (Pantalons vs Pulls) :  
t-statistique : 3.833197565497935, p-value : 0.0004170619187292952  
  
Test de Student (Robes vs Pulls) :  
t-statistique : 6.473790891367622, p-value : 0.0000000826
```

# Analyse des résultats:

1

## Pantalons vs Robes

p-value : **0.0004828**

La p-value est inférieure à 0.05, ce qui permet de **rejeter l'hypothèse nulle** et de conclure que les moyennes des ventes de pantalons et de robes sont **significativement différentes**.

t-statistique : **-3.78437**

Les pantalons ont une moyenne de ventes inférieure à celle des robes, comme l'indique la t-statistique négative.

2

## Pantalons vs Pulls

p-value : **0.0004170619**

La p-value est largement inférieure à 0.05, ce qui nous permet également de **rejeter l'hypothèse nulle** et de conclure que les moyennes des ventes de pantalons et de pulls sont **significativement différentes**.

t-statistique : **3.833197**

La t-statistique positive suggère que les pantalons ont des ventes supérieures à celles des pulls.

3

## Robes vs Pulls

p-value : **0.0000000826**

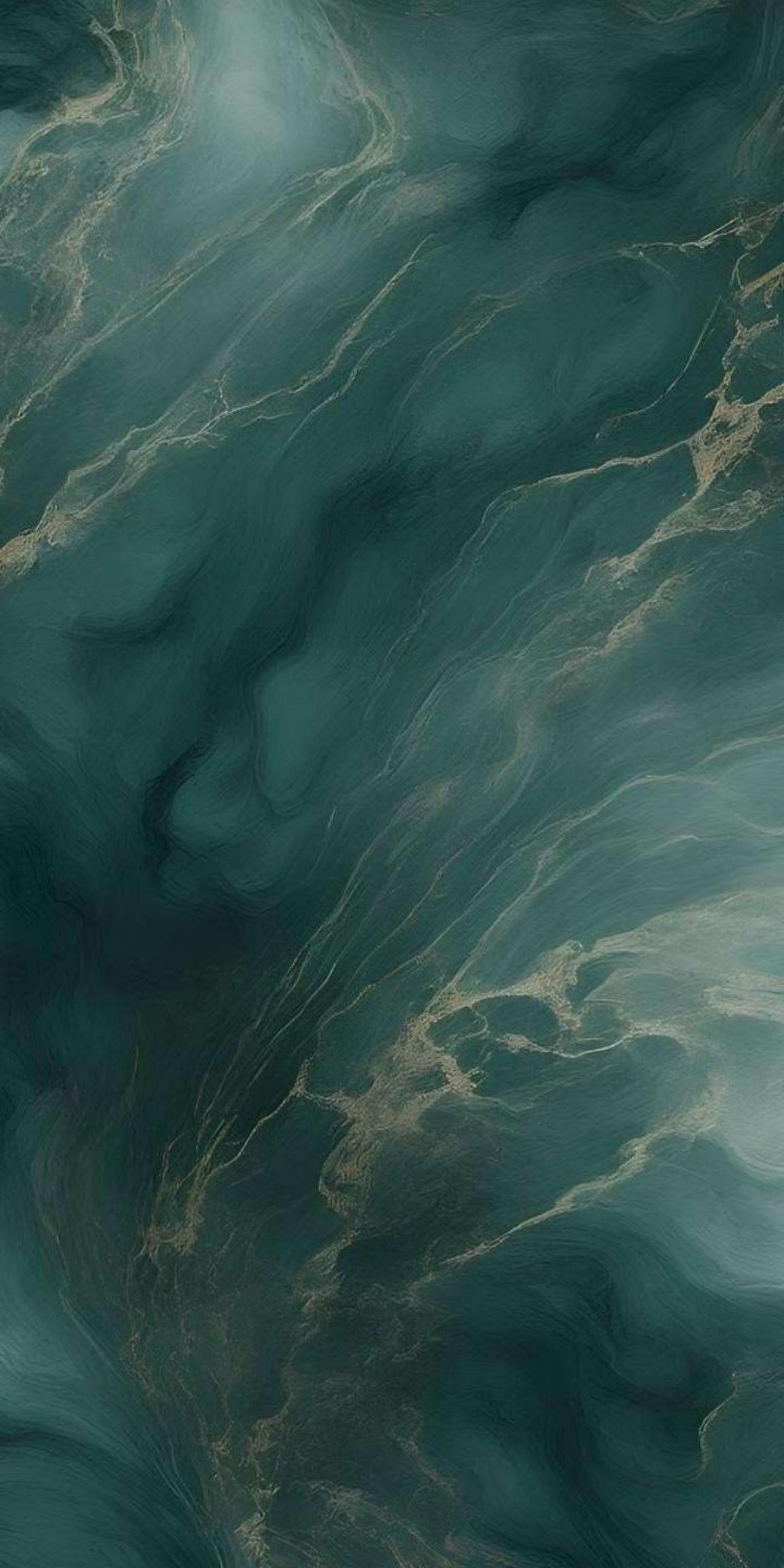
La p-value extrêmement faible (bien inférieure à 0.05) nous permet de **rejeter l'hypothèse nulle** et de conclure qu'il existe une différence **très significative** entre les moyennes des ventes de robes et de pulls.

t-statistique : **6.473790**

La t-statistique élevée montre que les ventes de robes sont beaucoup plus élevées que celles des pulls.

## 10—La réduction des données

:



# Analyse en Composantes Principales (ACP)

L'Analyse en Composantes Principales (ACP) est une **méthode statistique** puissante utilisée pour **réduire la dimensionnalité des données** tout en conservant l'essentiel de leur variabilité. Cette approche transforme les variables initiales en un ensemble de nouvelles variables non corrélées appelées **composants principaux**. Ces derniers sont ordonnés de manière à expliquer la plus grande partie de la variance des données.

Dans cette étude, l'ACP est appliquée pour réduire le nombre de fonctionnalités, ce qui permet d'optimiser l'analyse en réduisant la complexité tout en limitant la perte d'information. Une attention particulière est accordée au choix du nombre de composants principaux à retenir, en se basant sur des critères tels que la proportion de variance expliquée cumulée et l'analyse des valeurs propres.

# Analyse en Composantes Principales (ACP)

## Code:

```
# Fonction pour appliquer l'ACP
def apply_pca(df, name):
    # Sélectionner uniquement les colonnes numériques
    numeric_cols = df.select_dtypes(include=['float64', 'int64']).columns
    data = df[numeric_cols]

    # Standardisation des données
    scaler = StandardScaler()
    data_scaled = scaler.fit_transform(data)

    # Appliquer l'ACP
    pca = PCA()
    principal_components = pca.fit_transform(data_scaled)

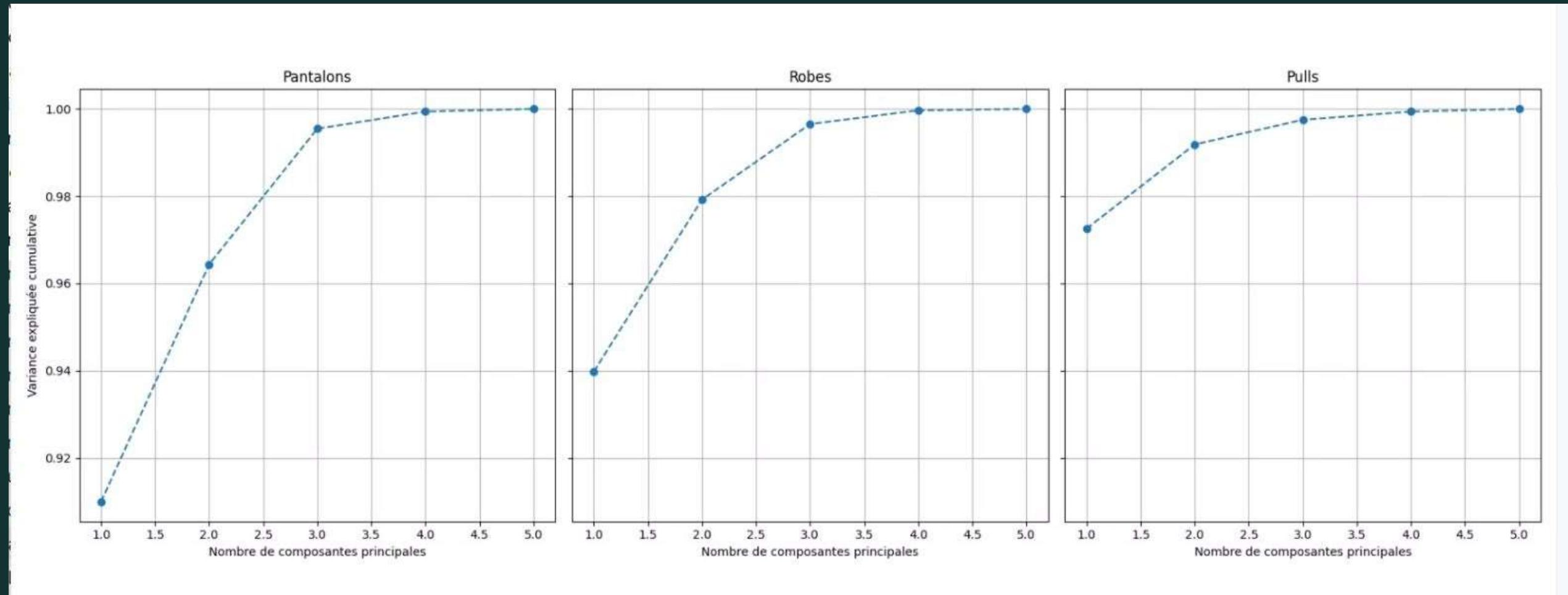
    # Variance expliquée
    explained_variance = pca.explained_variance_ratio_
    cumulative_variance = np.cumsum(explained_variance)

# Retourner les composantes principales et la variance expliquée cumulatif
return cumulative_variance, explained_variance

# Appliquer l'ACP sur chaque dataset nettoyé
pants_cumulative, pants_variance = apply_pca(df_pants_standardized, name: "Pantalons")
dress_cumulative, dress_variance = apply_pca(df_dress_standardized, name: "Robes")
sweater_cumulative, sweater_variance = apply_pca(df_sweater_standardized, name: "Pulls")
```

# Analyse en Composantes Principales (ACP)

## Analyse des Graphiques:



Ces graphes représentent la proportion de variance expliquée cumulée en fonction du nombre de composantes principales pour trois catégories différentes : **Pantalons**, **Robes**, et **Pulls**.

- Les **points** sur la courbe montrent la proportion cumulée de variance expliquée après l'ajout de chaque composante principale.
- La **courbe** illustre la progression de la variance expliquée lorsque des composantes supplémentaires sont incluses.

# Analyse en Composantes Principales (ACP)

## 1 Pantalons

Le graphique montre une augmentation rapide de la variance expliquée cumulée jusqu'à environ 2 composantes principales.

Après 2 composantes, l'augmentation de la variance expliquée devient beaucoup plus lente.

À 3 composantes, la variance expliquée cumulée est proche de 1 (ou 100%), indiquant que 3 composantes principales capturent presque toute la variance des données.

## 2 Robes

La variance expliquée cumulée augmente rapidement jusqu'à environ 2 composantes principales.

Après 2 composantes, l'augmentation devient plus lente.

À 3 composantes, la variance expliquée cumulée est très proche de 1, indiquant que 3 composantes principales capturent presque toute la variance des données.

## 3 Pulls

Le graphique montre une augmentation rapide de la variance expliquée cumulée jusqu'à environ 2 composantes principales.

Après 2 composantes, l'augmentation devient plus lente.

À 3 composantes, la variance expliquée cumulée est très proche de 1, indiquant que 3 composantes principales capturent presque toute la variance des données.

# Analyse en Composantes Principales (ACP)

## Discuter le choix du nombre de composants

Pour choisir le nombre optimal de composantes principales, il est important de trouver un équilibre entre la complexité du modèle et la quantité de variance expliquée.

1

**Variance Expliquée :** Pour les trois catégories, 3 composantes principales semblent capturer presque toute la variance des données (proche de 100%).

Cela suggère que 3 composantes principales sont suffisantes pour représenter les données de manière efficace.

2

**Complexité du Modèle :** Utiliser plus de 3 composantes principales n'apporterait qu'une augmentation marginale de la variance expliquée, tout en augmentant la complexité du modèle.

Utiliser moins de 3 composantes principales pourrait ne pas capturer suffisamment de variance, ce qui pourrait entraîner une perte d'information importante.

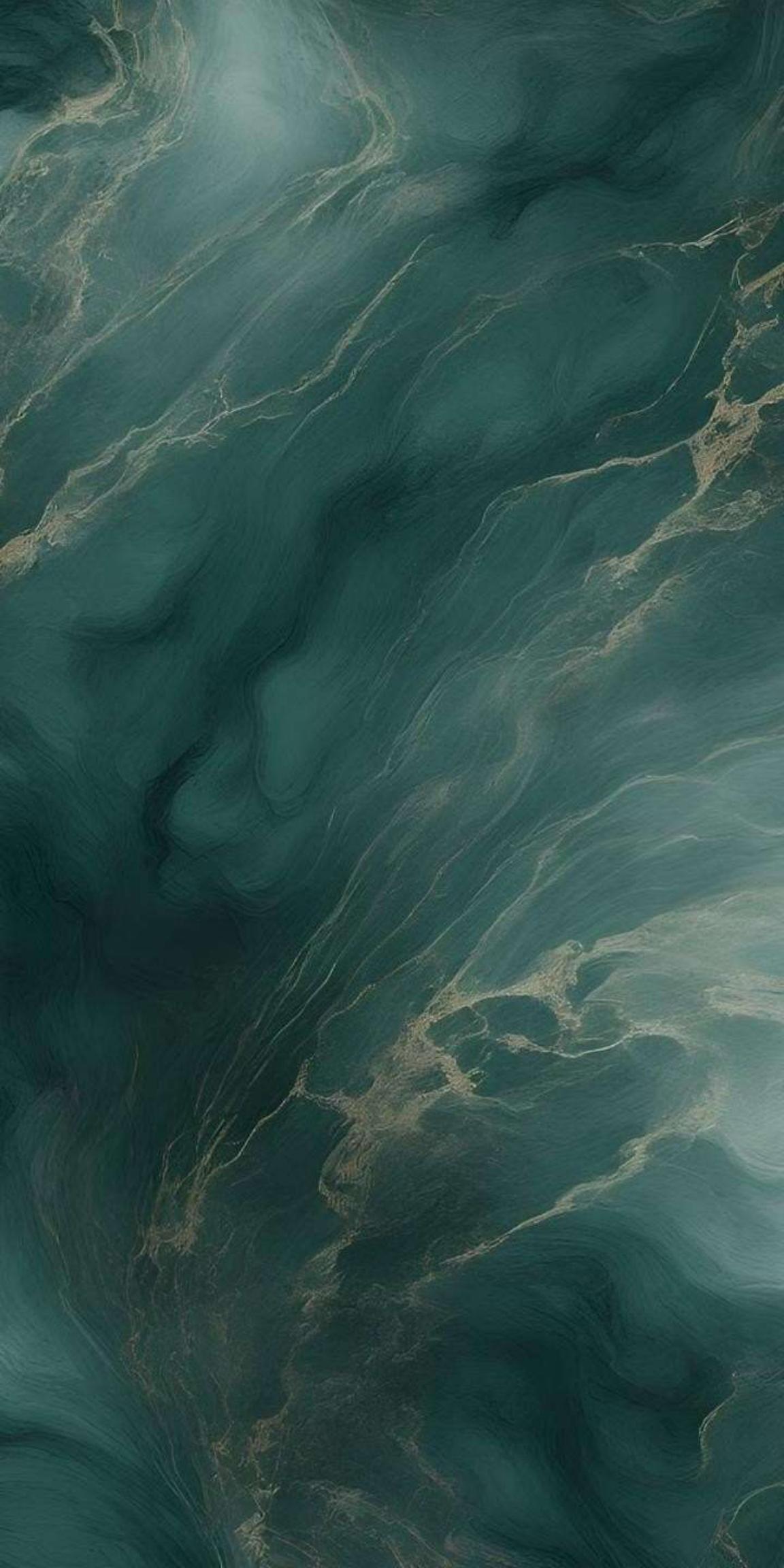
3

**Interprétabilité :** Un nombre réduit de composantes principales facilite l'interprétation des résultats.

Trois composantes principales offrent un bon compromis entre simplicité et représentation des données.

Pour les trois catégories de vêtements, le choix de 3 composantes principales semble être optimal. Ce choix permet de capturer presque toute la variance des données tout en maintenant une complexité du modèle raisonnable et en facilitant l'interprétation des résultats.

## 11—Prédiction des ventes



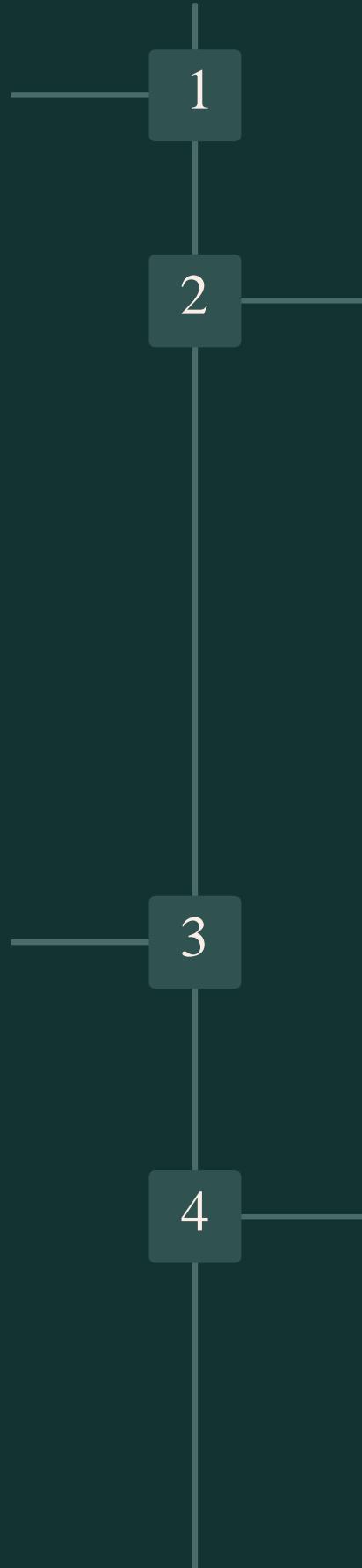
# Prédiction des ventes

## Division des données en ensembles d'entraînement et de test

- **L'ensemble d'entraînement :** utilisé pour ajuster le modèle.
- **L'ensemble de test :** utilisé pour évaluer la performance du modèle sur des données qu'il n'a jamais vues.
- Cette étape est essentielle pour garantir que le modèle peut généraliser correctement et éviter le surapprentissage.

## Test du modèle

Évaluer les performances du modèle sur l'ensemble de test en comparant les ventes prédictives aux ventes réelles. Cela permet de mesurer dans quelle mesure le modèle est capable de généraliser.



## Entraînement du modèle

Utiliser l'ensemble d'entraînement pour apprendre les relations entre les variables explicatives (caractéristiques) et la variable cible (ventes). Le modèle ajuste ses paramètres pour minimiser l'erreur de prédiction

## Visualisation des résultats sous forme de graphiques

Représenter graphiquement les prédictions par rapport aux valeurs réelles pour visualiser la qualité des prédictions.

# Prédiction des ventes

## Exemple du code pour pantalons :

```
# Importer les bibliothèques nécessaires
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import numpy as np

# Sélectionner les colonnes pertinentes pour l'entraînement
# Supposons que "S/ billion Yuan" soit la colonne cible (les ventes), et les autres colonnes sont les caractéristiques
# Ici, nous considérons des caractéristiques numériques uniquement
df_pants_features = df_pants_cleaned.drop(columns=["Date", "S/ billion Yuan"])
df_pants_target = df_pants_cleaned["S/ billion Yuan"]

# Étape 1 : Diviser les données en ensembles d'entraînement et de test
X_train, X_test, y_train, y_test = train_test_split(*arrays=df_pants_features, df_pants_target, test_size=0.2, random_state=42)

# Étape 2 : Entrainer le modèle de régression linéaire
model = LinearRegression()
model.fit(X_train, y_train)
# Étape 3 : Tester le modèle sur l'ensemble de test
y_pred = model.predict(X_test)

# Évaluer la performance du modèle
# Calculer l'erreur quadratique moyenne (RMSE)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))

print("Erreur quadratique moyenne (RMSE) : ", rmse)

# Afficher les coefficients du modèle pour chaque caractéristique
print("Coefficients du modèle : ", model.coef_)

# Afficher l'ordonnée à l'origine (intercept)
print("Ordonnée à l'origine (intercept) : ", model.intercept_)

# Comparer les valeurs réelles et prédictives pour un échantillon
comparison = pd.DataFrame({"Ventes réelles": y_test, "Ventes prédictes": y_pred})
print(comparison.head())
```

# Prédiction des ventes

## Analyse des résultats du code :

### 1-Pantalons

```
Erreur quadratique moyenne (RMSE) pour les pulls : 0.13714513989736002
Coefficients du modèle pour les pulls : [-0.25489596  0.12042262 -0.06791396  0.24940899]
Ordonnée à l'origine (intercept) pour les pulls : -0.6781138617571572
      Ventes réelles   Ventes prédictes
35            1.15          1.248759
4             1.90          2.057948
8             1.27          1.224326
13            0.55          0.495397
15            1.27          1.502971
```

Le modèle présente une **bonne précision** avec un **RMSE de 0.0858**, ce qui indique un faible écart moyen entre les valeurs réelles et prédictes des ventes.

Les **coefficients du modèle** montrent que certaines variables ont un effet positif (ex : 0.44 pour la deuxième variable), tandis que d'autres ont un effet négatif (ex : -0.58 pour la troisième variable). L'**intercept** de -1.48 représente la valeur prédictée lorsque toutes les variables sont nulles.

La comparaison entre les **ventes réelles** et **prédictes** montre que le modèle prédit est assez bien , avec des différences minimales, ce qui suggère une **prédiction fiable**.

# Prédiction des ventes

## 2-Robes

```
Erreur quadratique moyenne (RMSE) pour les robes : 0.1309513722731203
```

```
Coefficients du modèle pour les robes : [ 0.02608563  0.04048354 -0.51855373  4.0627336 ]
```

```
Ordonnée à l'origine (intercept) pour les robes : -2.575766544437193
```

	Ventes réelles	Ventes prédictes
17	2.62	2.611187
32	3.30	3.391521
24	4.71	4.896565
13	6.85	6.790417
19	1.81	1.612691

Le modèle pour les **robes** a un **RMSE de 0.1310**, ce qui indique une bonne précision.

Les **coefficients** montrent que la quatrième variable a un fort effet positif, tandis que la troisième a un effet négatif.

L'**intercept** est de -2.5758.

Les **ventes réelles** et **prédictes** sont proches, avec des différences faibles, notamment pour les points 17, 32, et 24, bien qu'il y ait une légère sous-estimation pour le point 19.

# Prédiction des ventes

## 3-Pulls

```
Erreur quadratique moyenne (RMSE) pour les pulls : 0.13714513989736002
Coefficients du modèle pour les pulls : [-0.25489596  0.12042262 -0.06791396  0.24940899]
Ordonnée à l'origine (intercept) pour les pulls : -0.6781138617571572
      Ventes réelles   Ventes prédictes
35          1.15        1.248759
4           1.90        2.057948
8           1.27        1.224326
13          0.55        0.495397
15          1.27        1.502971
```

Le modèle pour les **pulls** a un **RMSE de 0.1371**, indiquant un écart modéré entre les ventes réelles et prédictes.

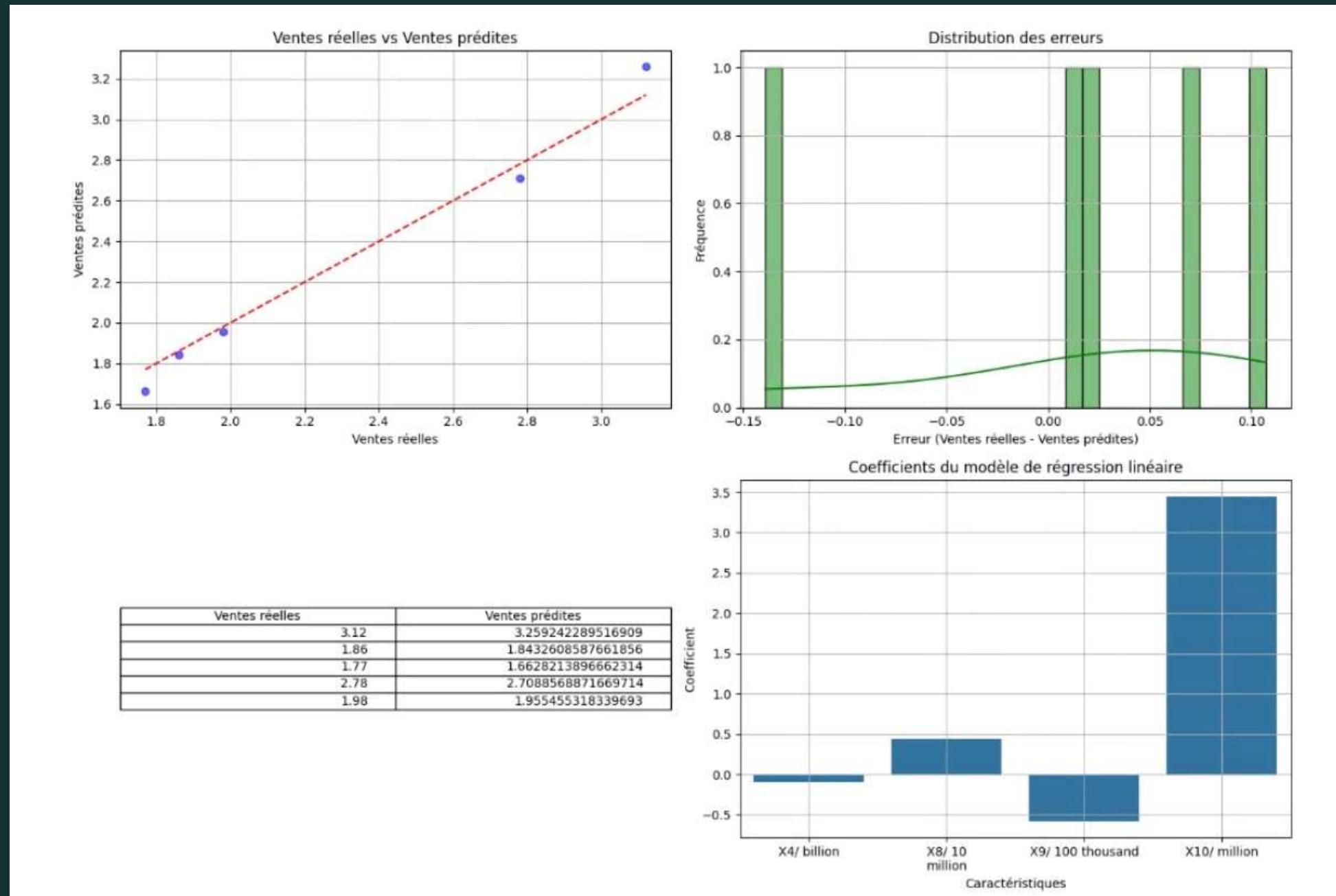
Les **coefficients** montrent des effets mixtes : certaines variables influencent positivement les ventes, tandis que d'autres ont un effet négatif. L'**intercept** est de -0.6781.

La comparaison des **ventes réelles et prédictes** montre que les prédictions sont généralement proches des valeurs réelles, bien que quelques différences apparaissent, notamment pour les points 4 et 15.

# Prédiction des ventes

## Visualisation des graphiques:

### 1-Pantalons:



# Prédiction des ventes

## 1-Pantalons:

### 1 Ventes réelles vs Ventes prédictes

Ce graphique montre la relation entre les ventes réelles et les ventes prédictes pour les pantalons. Les points bleus représentent les données réelles, et la ligne rouge pointillée représente la ligne de régression linéaire parfaite où les ventes prédictes seraient égales aux ventes réelles.

Les points sont proches de la ligne rouge, ce qui indique que le modèle de prédiction est assez précis.

Il y a une légère dispersion autour de la ligne, ce qui montre que certaines prédictions sont légèrement en dessous ou au-dessus des ventes réelles.

Les points sont bien alignés le long de la ligne, ce qui suggère une bonne adéquation du modèle.

### 2 Distribution des erreurs

Ce graphique montre la distribution des erreurs de prédiction (ventes réelles - ventes prédictes).

La majorité des erreurs sont proches de zéro, ce qui est un bon signe car cela signifie que les prédictions sont proches des valeurs réelles.

Les erreurs sont symétriquement distribuées autour de zéro, indiquant que le modèle ne sur-prédit ni ne sous-prédict de manière systématique.

La fréquence des erreurs diminue à mesure que l'on s'éloigne de zéro, ce qui montre que les grandes erreurs sont rares.

### 3 Coefficients du modèle de régression linéaire

Ce graphique montre les coefficients des différentes caractéristiques utilisées dans le modèle de régression linéaire.

La caractéristique X10 (million) a le coefficient le plus élevé, indiquant qu'elle a la plus grande influence positive sur les ventes prédictes.

La caractéristique X9 (100 thousand) a un coefficient négatif, ce qui signifie qu'elle a un impact négatif sur les ventes prédictes.

Les caractéristiques X4 (billion) et X8 (10 million) ont des coefficients positifs mais plus faibles, indiquant une influence modérée sur les prédictions.

### 4 Tableau des ventes réelles vs ventes prédictes

Ce tableau compare les ventes réelles et les ventes prédictes pour différentes observations.

Les valeurs prédictes sont très proches des valeurs réelles, ce qui confirme la précision du modèle.

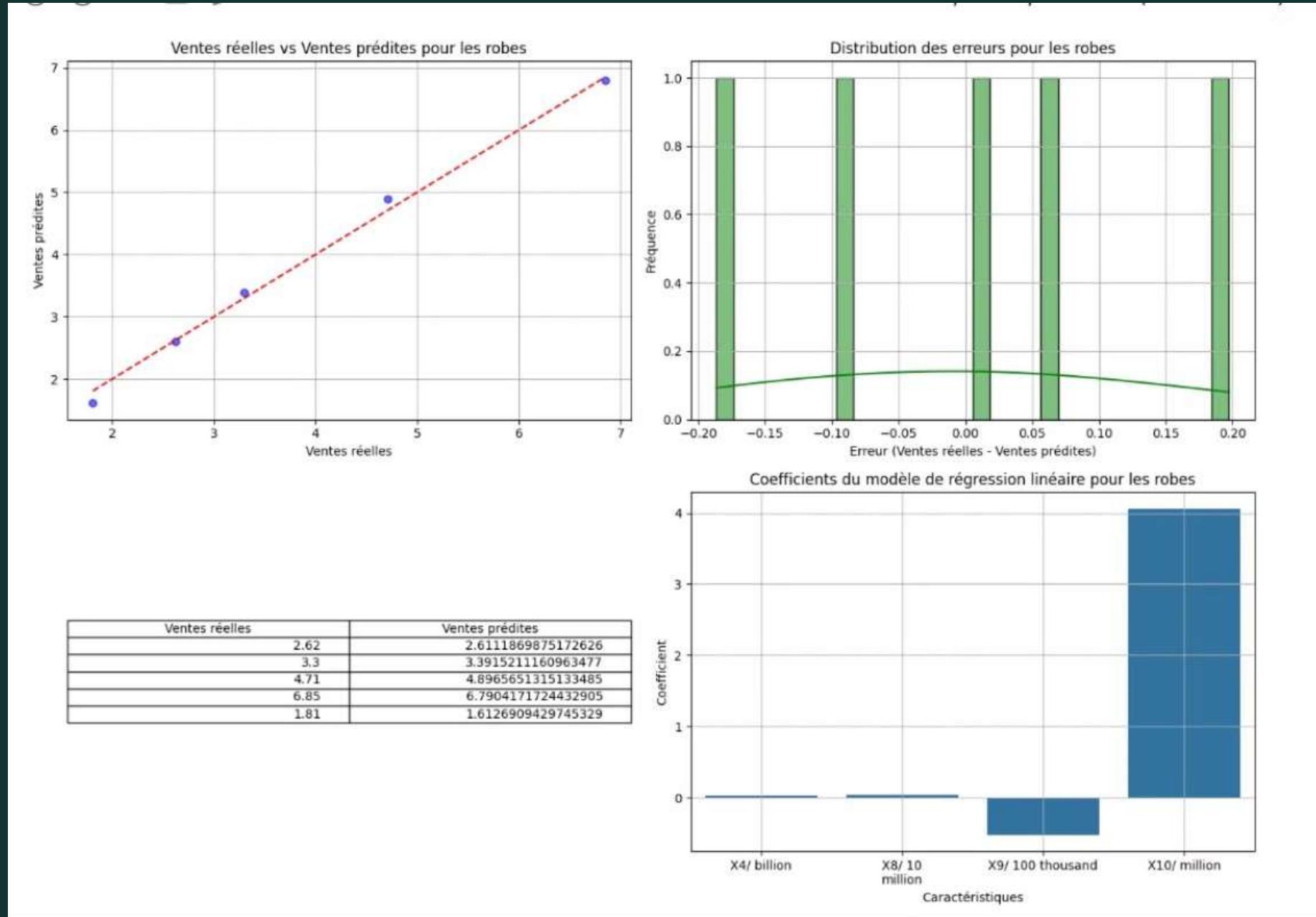
Par exemple, pour des ventes réelles de 3.12, la prédiction est de 3.25, ce qui montre une très bonne précision.

Les écarts entre les valeurs réelles et prédictes sont minimes, ce qui renforce la fiabilité du modèle.

En résumé, le modèle de prédiction des ventes semble être très précis. Les graphiques montrent que les prédictions sont proches des valeurs réelles, avec des erreurs symétriquement distribuées autour de zéro. Le modèle de régression linéaire utilisé semble bien ajusté, avec la caractéristique X10 ayant un impact significatif positif sur les prédictions, tandis que X9 a un impact négatif.

# Prédiction des ventes

## 2-Robes



# Prédiction des ventes

## 2-Robes

1

### Ventes réelles vs Ventes prédictes

Le graphique montre la relation entre les ventes réelles et les ventes prédictes pour les robes. Les points bleus représentent les données réelles, et la ligne rouge pointillée représente la ligne de régression linéaire parfaite où les ventes prédictes seraient égales aux ventes réelles.

Les points sont proches de la ligne rouge, ce qui indique que le modèle de prédiction est assez précis. Il y a une légère dispersion autour de la ligne, ce qui montre que certaines prédictions sont légèrement en dessous ou au-dessus des ventes réelles.

2

### Distribution des erreurs

Ce graphique montre la distribution des erreurs de prédiction (ventes réelles - ventes prédictes).

La majorité des erreurs sont proches de zéro, ce qui est un bon signe car cela signifie que les prédictions sont proches des valeurs réelles. Les erreurs sont symétriquement distribuées autour de zéro, indiquant que le modèle ne sur-prédit ni ne sous-préduit de manière systématique.

3

### Coefficients du modèle de régression linéaire

Ce graphique montre les coefficients des différentes caractéristiques utilisées dans le modèle de régression linéaire.

La caractéristique X10 (million) a le coefficient le plus élevé, indiquant qu'elle a la plus grande influence sur les ventes prédictes. Les autres caractéristiques (X4, X8, X9) ont des coefficients beaucoup plus faibles, ce qui signifie qu'elles ont moins d'impact sur les prédictions.

En résumé, le modèle de prédiction des ventes pour les robes semble être très précis. Les graphiques montrent que les prédictions sont proches des valeurs réelles, avec des erreurs symétriquement distribuées autour de zéro. Le modèle de régression linéaire utilisé semble bien ajusté, avec une caractéristique (X10) ayant un impact significatif sur les prédictions.

4

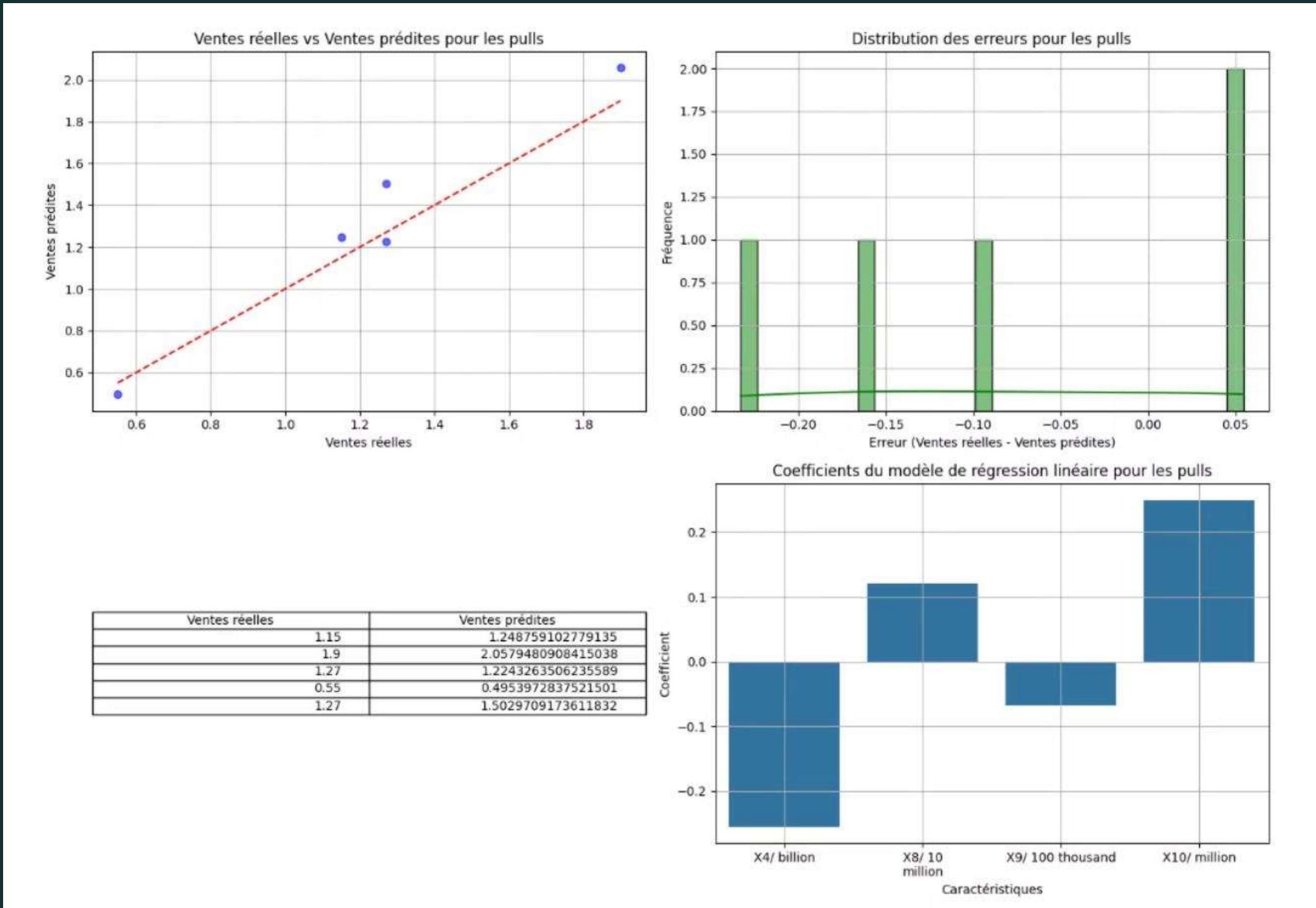
### Tableau des ventes réelles vs ventes prédictes

Ce tableau compare les ventes réelles et les ventes prédictes pour différentes observations.

Les valeurs prédictes sont très proches des valeurs réelles, ce qui confirme la précision du modèle. Par exemple, pour des ventes réelles de 2.62, la prédiction est de 2.61, ce qui montre une très bonne précision.

# Prédiction des ventes

## 3-Pulls



# Prédiction des ventes

## 3-Pulls

1

### Ventes réelles vs Ventes prédictes

Ce graphique montre la relation entre les ventes réelles et les ventes prédictes pour les pulls. Les points bleus représentent les données réelles, et la ligne rouge pointillée représente la ligne de régression linéaire parfaite où les ventes prédictes seraient égales aux ventes réelles.

Les points sont relativement proches de la ligne rouge, ce qui indique que le modèle de prédiction est assez précis.

Il y a une certaine dispersion autour de la ligne, ce qui montre que certaines prédictions sont légèrement en dessous ou au-dessus des ventes réelles.

Un point semble être un peu plus éloigné de la ligne, ce qui pourrait indiquer une prédiction moins précise pour cette observation particulière.

2

### Distribution des erreurs

Ce graphique montre la distribution des erreurs de prédiction (ventes réelles - ventes prédictes).

La majorité des erreurs sont proches de zéro, ce qui est un bon signe car cela signifie que les prédictions sont proches des valeurs réelles.

Les erreurs sont symétriquement distribuées autour de zéro, indiquant que le modèle ne sur-prédit ni ne sous-préduit de manière systématique.

La fréquence des erreurs diminue à mesure que l'on s'éloigne de zéro, ce qui montre que les grandes erreurs sont rares.

3

### Coefficients du modèle de régression linéaire

Ce graphique montre les coefficients des différentes caractéristiques utilisées dans le modèle de régression linéaire.

La caractéristique X10 (million) a le coefficient le plus élevé, indiquant qu'elle a la plus grande influence positive sur les ventes prédictes.

La caractéristique X4 (billion) a un coefficient négatif, ce qui signifie qu'elle a un impact négatif sur les ventes prédictes.

Les caractéristiques X8 (10 million) et X9 (100 thousand) ont des coefficients positifs mais plus faibles, indiquant une influence modérée sur les prédictions.

4

### Tableau des ventes réelles vs ventes prédictes

Ce tableau compare les ventes réelles et les ventes prédictes pour différentes observations.

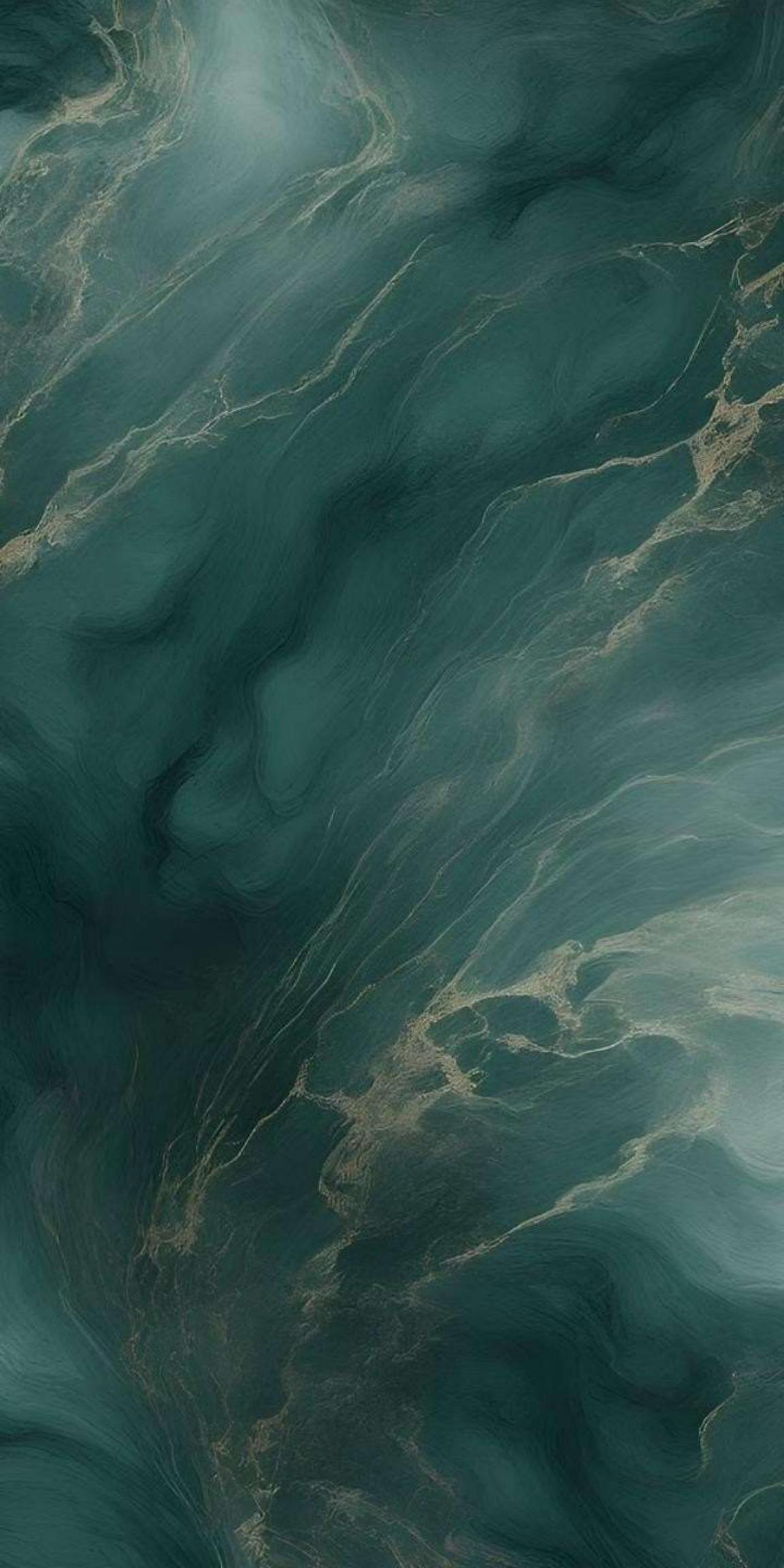
Les valeurs prédictes sont assez proches des valeurs réelles, ce qui confirme la précision du modèle.

Par exemple, pour des ventes réelles de 1.15, la prédiction est de 1.24, ce qui montre une bonne précision.

Les écarts entre les valeurs réelles et prédictes sont généralement minimes, ce qui renforce la fiabilité du modèle.

En résumé, le modèle de prédiction des ventes pour les pulls semble être assez précis. Les graphiques montrent que les prédictions sont proches des valeurs réelles, avec des erreurs symétriquement distribuées autour de zéro. Le modèle de régression linéaire utilisé semble bien ajusté, avec la caractéristique X10 ayant un impact significatif positif sur les prédictions, tandis que X4 a un impact négatif.

# Conclusion



# Conclusion

Ce projet d'analyse des données de ventes a permis d'optimiser les performances commerciales des catégories de produits vestimentaires (pantalons, robes et pulls) en utilisant des techniques statistiques avancées. Après la collecte et la préparation des données, incluant la gestion des valeurs manquantes et la suppression des doublons, les valeurs aberrantes ont été détectées et traitées. La standardisation des données a homogénéisé les variables pour une analyse précise. Les relations et corrélations entre les caractéristiques et les ventes ont été visualisées, révélant des tendances significatives. Les tests de Student ont confirmé des différences notables entre les moyennes des ventes des différentes catégories. L'Analyse en Composantes Principales (ACP) a réduit la dimensionnalité des données tout en conservant leur variabilité essentielle. Enfin, des modèles de régression linéaire ont été développés pour prédire les ventes futures avec une bonne précision, fournissant ainsi des outils précieux pour la gestion des stocks et la prise de décisions stratégiques.

*Merci pour votre attention !*

