Mohammed V University

Faculty Of Sciences Rabat
2024/2025

Title: Analyzing and Segmenting Agricultural Product Prices in Morocco Using Clustering Algorithms

Program: Msc Intelligent Processing Systems
Course: Data Mining
Student : CHAOUCHAOU Omaima
AIT EL MOUDEN Khaoula
Lecturer : Pf. RIAD SOLH Anouar

Abstract: This project focuses on the analysis and segmentation of agricultural product prices in Morocco using various clustering algorithms. The goal is to uncover patterns in pricing and segment products based on price similarities, helping to understand price dynamics in the agricultural market. Initially, the data was cleaned by removing missing values, handling outliers, and encoding categorical variables. We then employed multiple clustering techniques, such as K-Means, DBSCAN, and KNN, alongside Principal Component Analysis (PCA) for dimensionality reduction to explore the data. We also applied the Apriori algorithm for association rule mining to discover relationships between different products and their pricing trends. The performance of the models was evaluated using accuracy scores and visualized through various graphical representations such as bar charts, scatter plots, and histograms. This project provides valuable insights into the agricultural pricing landscape in Morocco, with potential applications in pricing strategies, market forecasting, and policy formulation.

Keywords:
Agricultural pricing, Clustering, K-Means, DBSCAN, PCA,KNN, Apriori algorithm, Price segmentation, Morocco.

1 Introduction

Agriculture is a key sector in Morocco's economy, with product prices influenced by various factors like region, season, and supply-demand. Understanding these price fluctuations is essential for farmers, policymakers, and consumers. This project focuses on analyzing and segmenting agricultural product prices in Morocco using clustering algorithms such as K-Means, DBSCAN, and KNN. Principal Component Analysis (PCA) is applied to reduce data dimensionality, while the Apriori algorithm helps uncover relationships between products and pricing trends. Ultimately, this study aims to provide valuable insights into the agricultural pricing landscape, aiding in pricing strategies and market forecasting.

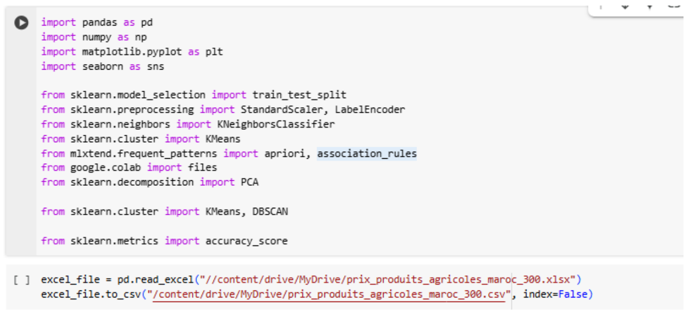
2 Why This Topic?

The choice of analyzing agricultural product prices in Morocco stems from the importance of agriculture in the country's economy. Agricultural products such as cereals, fruits, and vegetables play a crucial role in both local consumption and export markets. By studying these prices, we can gain insights into market trends, identify factors influencing price fluctuations, and potentially guide decision-making for both producers and consumers. Additionally, this dataset offers an opportunity to apply advanced machine learning techniques such as clustering, classification, and association rule mining to real-world problems in the agricultural sector. The data used in this analysis was sourced from the Kaggle dataset titled "Prix des Produits Agricoles au Maroc", and the analysis was conducted using Google Colab, a powerful tool for collaborative Python-based coding and data analysis.

3 Data Analytics

1. Importing the Dataset

Before any analysis can begin, we need to import the data from an external file (in this case, an Excel file). After that, we convert it to a more accessible format, such as CSV, for easier manipulation and future use.



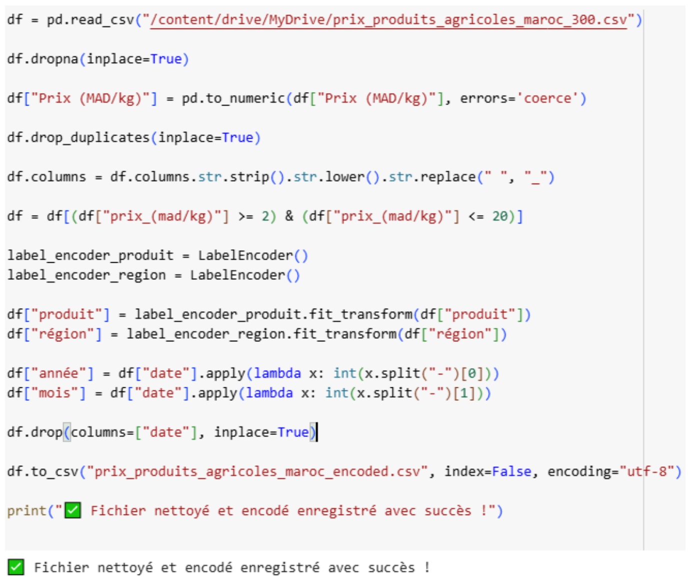
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.neighbors import KNeighborsClassifier
from sklearn.cluster import KMeans
from mlxtend.frequent_patterns import apriori, association_rules
from google.colab import files
from sklearn.decomposition import PCA

from sklearn.cluster import KMeans, DBSCAN

from sklearn.metrics import accuracy_score
```

```
[ ] excel_file = pd.read_excel("//content/drive/MyDrive/prix_produits_agricoles_maroc_300.xlsx")
    excel_file.to_csv("/content/drive/MyDrive/prix_produits_agricoles_maroc_300.csv", index=False)
```

- **Importing necessary libraries**: We import libraries like pandas for data manipulation, matplotlib/seaborn for visualization, and other machine learning tools (e.g., KNeighborsClassifier, KMeans) for later use.
- **Google Colab Drive mount**: drive. mount ('/content/drive') allows us to access files stored in Google Drive (used when working in Google Colab).
- **Loading the Excel file**: The Excel file is loaded into a DataFrame using pd.Read_excel (). This makes it easy to manipulate the data in Python.
- **Saving as a CSV file**: We convert the Excel file to CSV using `to_csv()`. This step makes the file easier to work with if needed for further processing.

2. Data Cleaning

Data cleaning is a crucial step in data analysis and machine learning, transforming raw data into a usable format. It ensures the dataset is free from errors, inconsistencies, and missing values, improving accuracy. The goal is to create a consistent and reliable dataset, ready for further analysis.



```
df = pd.read_csv("/content/drive/MyDrive/prix_produits_agricoles_maroc_300.csv")

df.dropna(inplace=True)

df["Prix (MAD/kg)"] = pd.to_numeric(df["Prix (MAD/kg)"], errors='coerce')

df.drop_duplicates(inplace=True)

df.columns = df.columns.str.strip().str.lower().str.replace(" ", "_")

df = df[(df["prix_(mad/kg)"] >= 2) & (df["prix_(mad/kg)"] <= 20)]

label_encoder_produit = LabelEncoder()
label_encoder_region = LabelEncoder()

df["produit"] = label_encoder_produit.fit_transform(df["produit"])
df["région"] = label_encoder_region.fit_transform(df["région"])

df["année"] = df["date"].apply(lambda x: int(x.split("-")[0]))
df["mois"] = df["date"].apply(lambda x: int(x.split("-")[1]))

df.drop(columns=["date"], inplace=True)

df.to_csv("prix_produits_agricoles_maroc_encoded.csv", index=False, encoding="utf-8")

print("✅ Fichier nettoyé et encodé enregistré avec succès !")
```

```
✅ Fichier nettoyé et encodé enregistré avec succès !
```

Data cleaning involves several key steps to prepare the data for analysis:

- **Missing Data**: Any rows containing missing values are removed to ensure the dataset is complete.
- **Type Conversion**: The price column is converted to numeric format to allow for proper analysis and filtering.
- **Duplicate Removal**: Duplicate rows are removed to ensure the dataset contains only unique entries.
- **Column Name Standardization**: The column names are standardized to lower case and spaces are replaced with underscores for consistency.
- **Outlier Removal**: Any data points with unrealistic prices (outside a specified range) are filtered out to ensure the dataset reflects reasonable values.

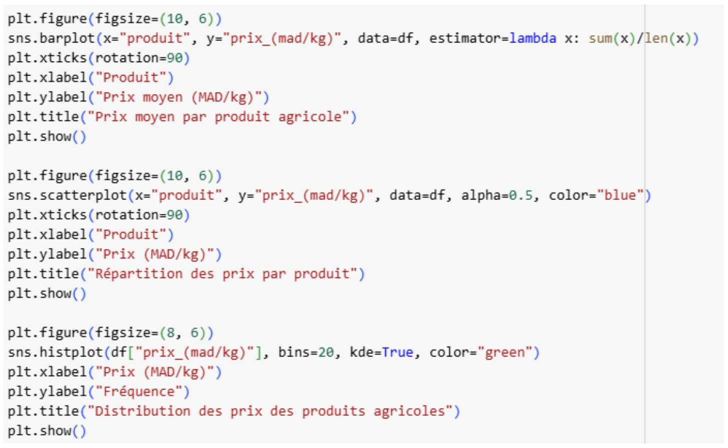
3. Categorical Data Encoding

Since some columns contain categorical data (such as product and region names), they need to be converted into numeric values for analysis. This is done using a label encoding method, which assigns a unique numeric value to each category.
LabelEncoder is used to convert categorical columns into numeric values.
Each unique category within the "produit" and "région" columns is assigned a unique integer, making the data ready for analysis or machine learning algorithms.

4. Data Visualization of Agricultural Product Prices

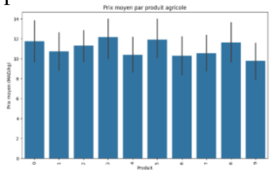
Visualization of Agricultural Product Prices: Comparison, Distribution, and Variability

The visualizations aim to provide a comprehensive overview of agricultural product prices by highlighting the average price per product, the distribution of prices within each product, and the overall price distribution. This helps identify key pricing trends and fluctuations, offering valuable insights for farmers, policymakers, and market analysts to make informed decisions.

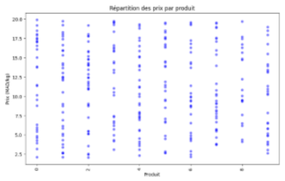


```
plt.figure(figsize=(10, 6))
sns.barplot(x="produit", y="prix_(mad/kg)", data=df, estimator=lambda x: sum(x)/len(x))
plt.xticks(rotation=90)
plt.xlabel("Produit")
plt.ylabel("Prix moyen (MAD/kg)")
plt.title("Prix moyen par produit agricole")
plt.show()

plt.figure(figsize=(10, 6))
sns.scatterplot(x="produit", y="prix_(mad/kg)", data=df, alpha=0.5, color="blue")
plt.xticks(rotation=90)
plt.xlabel("Produit")
plt.ylabel("Prix (MAD/kg)")
plt.title("Répartition des prix par produit")
plt.show()

plt.figure(figsize=(8, 6))
sns.histplot(df["prix_(mad/kg)"], bins=20, kde=True, color="green")
plt.xlabel("Prix (MAD/kg)")
plt.ylabel("Fréquence")
plt.title("Distribution des prix des produits agricoles")
plt.show()
```

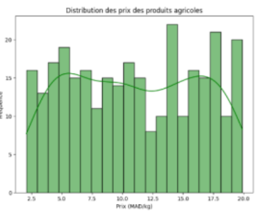
The bar chart compares the average prices of agricultural products, with taller bars indicating higher prices. Error bars represent price variability, showing how consistent the pricesare within each product.



The scatter plot shows the price variation within each product category, with dots representing individual price points. It reveals price clustering, outliers, and overall price ranges, but lacks product labels, making it difficult to identify specific products.



This histogram with a KDE curve shows the distribution of agricultural product prices. It reveals the central price range, variability, and potential patterns, such as price clustering around 10-15 MAD/kg. The KDE curve smooths out the data, offering a clearer view of the overall price distribution.



- **Bar Chart** : Compares average prices across products with error bars, showing reliability. Adding labels and considering price variability would enhance the analysis.

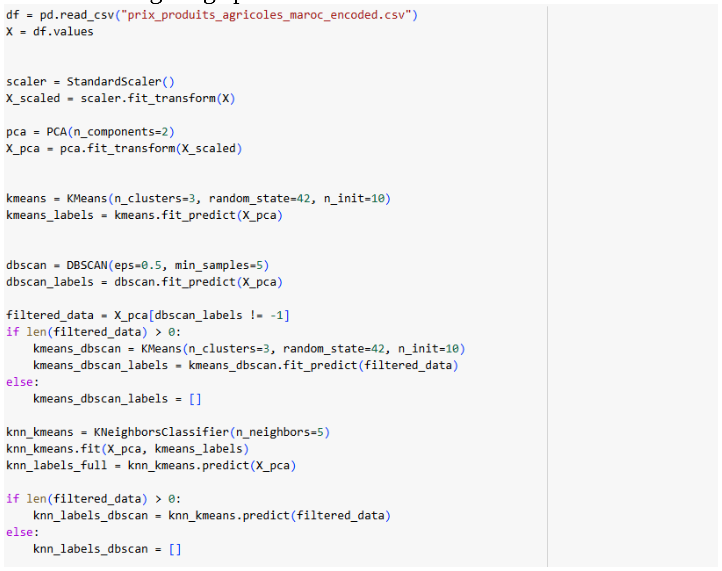
- **Histogram with KDE** : Visualizes price distribution, revealing central tendencies and trends. It highlights patterns in agricultural product prices.

- **Scatter Plot** : Shows price distribution by product. Adding labels and combining with other plots would improve clarity and provide more insights.

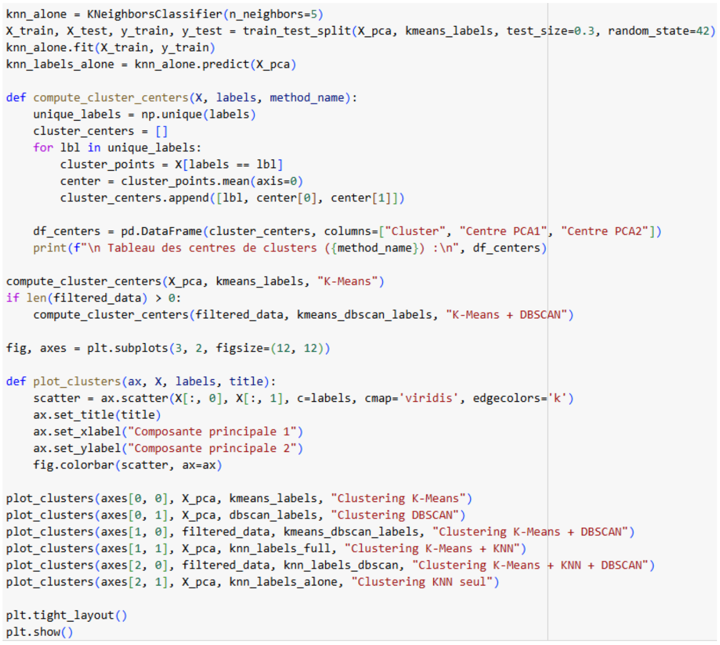
4 Application of Modeling Algorithms

1. Clustering Analysis

This analysis applies various clustering algorithms to explore agricultural product prices in Morocco, using PCA for dimensionality reduction to create a 2D representation. It employs K-Means, DBSCAN, and combinations of K-Means with DBSCAN and KNN to refine clusters and classify data points. The goal is to uncover pricing patterns and identify potential market segments. Visualizations and cluster center calculations help pinpoint significant price groupings. The insights gained aid decision-makers in understanding pricing trends and targeting specific markets.

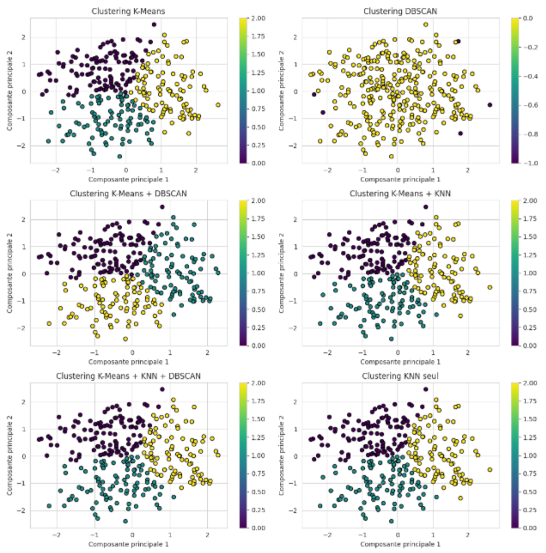


```
df = pd.read_csv("prix_produits_agricoles_maroc_encoded.csv")
X = df.values


scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)


kmeans = KMeans(n_clusters=3, random_state=42, n_init=10)
kmeans_labels = kmeans.fit_predict(X_pca)


dbscan = DBSCAN(eps=0.5, min_samples=5)
dbscan_labels = dbscan.fit_predict(X_pca)

filtered_data = X_pca[dbscan_labels != -1]
if len(filtered_data) > 0:
    kmeans_dbscan = KMeans(n_clusters=3, random_state=42, n_init=10)
    kmeans_dbscan_labels = kmeans_dbscan.fit_predict(filtered_data)
else:
    kmeans_dbscan_labels = []

knn_kmeans = KNeighborsClassifier(n_neighbors=5)
knn_kmeans.fit(X_pca, kmeans_labels)
knn_labels_full = knn_kmeans.predict(X_pca)

if len(filtered_data) > 0:
    knn_labels_dbscan = knn_kmeans.predict(filtered_data)
else:
    knn_labels_dbscan = []
```

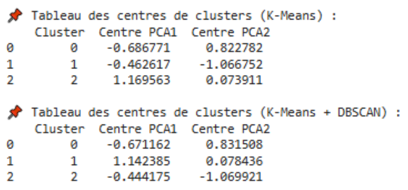


```
knn_alone = KNeighborsClassifier(n_neighbors=5)
X_train, X_test, y_train, y_test = train_test_split(X_pca, kmeans_labels, test_size=0.3, random_state=42)
knn_alone.fit(X_train, y_train)
knn_labels_alone = knn_alone.predict(X_pca)

def compute_cluster_centers(X, labels, method_name):
    unique_labels = np.unique(labels)
    cluster_centers = []
    for lbl in unique_labels:
        cluster_points = X[labels == lbl]
        center = cluster_points.mean(axis=0)
        cluster_centers.append([lbl, center[0], center[1]])

    df_centers = pd.DataFrame(cluster_centers, columns=["Cluster", "Centre PCA1", "Centre PCA2"])
    print(f"\n Tableau des centres de clusters ({method_name}) :\n", df_centers)

compute_cluster_centers(X_pca, kmeans_labels, "K-Means")
if len(filtered_data) > 0:
    compute_cluster_centers(filtered_data, kmeans_dbscan_labels, "K-Means + DBSCAN")

fig, axes = plt.subplots(3, 2, figsize=(12, 12))

def plot_clusters(ax, X, labels, title):
    scatter = ax.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis', edgecolors='k')
    ax.set_title(title)
    ax.set_xlabel("Composante principale 1")
    ax.set_ylabel("Composante principale 2")
    fig.colorbar(scatter, ax=ax)

plot_clusters(axes[0, 0], X_pca, kmeans_labels, "Clustering K-Means")
plot_clusters(axes[0, 1], X_pca, dbscan_labels, "Clustering DBSCAN")
plot_clusters(axes[1, 0], filtered_data, kmeans_dbscan_labels, "Clustering K-Means + DBSCAN")
plot_clusters(axes[1, 1], X_pca, knn_labels_full, "Clustering K-Means + KNN")
plot_clusters(axes[2, 0], filtered_data, knn_labels_dbscan, "Clustering K-Means + KNN + DBSCAN")
plot_clusters(axes[2, 1], X_pca, knn_labels_alone, "Clustering KNN seul")

plt.tight_layout()
plt.show()
```

- The code starts with **loading, scaling, and reducing dimensions** of the dataset (Step 1).
- Then, **multiple clustering methods** are applied, including K-Means, DBSCAN, and combinations of K-Means with KNN and DBSCAN (Step 2).
- Finally, **cluster centers** are computed and **visualizations** are generated to display the results (Step 3).



The visualizations of different clustering methods in PCA space revealed that K-Means identified roughly circular clusters, while DBSCAN found core clusters and noise. Combining K-Means with DBSCAN resulted in more compact and well-defined clusters, confirming the refinement of the cluster centers. The K-Means + KNN and KNN-alone methods showed similar patterns, reinforcing the overall clustering structure.



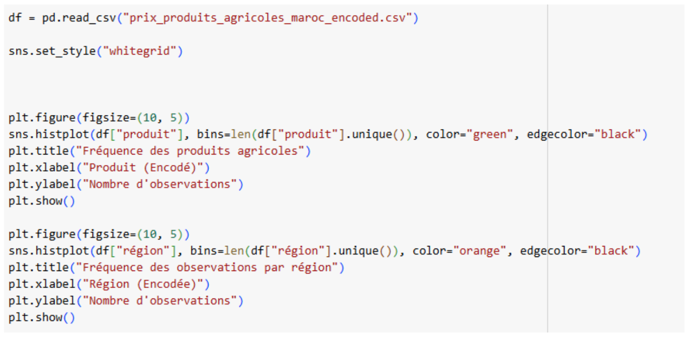
```
📌 Tableau des centres de clusters (K-Means) :
   Cluster  Centre PCA1  Centre PCA2
0        0    -0.686771     0.822782
1        1    -0.462617    -1.066752
2        2     1.169563     0.073911

📌 Tableau des centres de clusters (K-Means + DBSCAN) :
   Cluster  Centre PCA1  Centre PCA2
0        0    -0.671162     0.831508
1        1     1.142385     0.078436
2        2    -0.444175    -1.069921
```

From our clustering analysis, we observe that K-Means and K-Means + DBSCAN both identified three clusters, but the addition of DBSCAN led to slightly more refined results by removing outliers. The cluster centers in both methods are very similar, indicating that DBSCAN's main contribution was noise reduction rather than altering the fundamental structure of the data.

The clustering analysis effectively segmented agricultural products based on pricing characteristics, with K-Means and DBSCAN providing refined and meaningful clusters. This segmentation offers valuable insights for policymakers and farmers to optimize pricing strategies. Further exploration of original product features and the optimal number of clusters is recommended for deeper understanding.

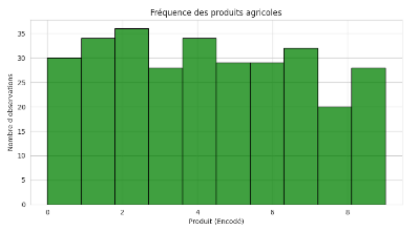
2. Distribution Analysis

This script generates histograms to visualize the distribution of two categorical variables: "produit" (product) and "région" (region) from the dataset. It helps to identify the frequency of agricultural products and regional observations, providing insights into the dataset's composition.

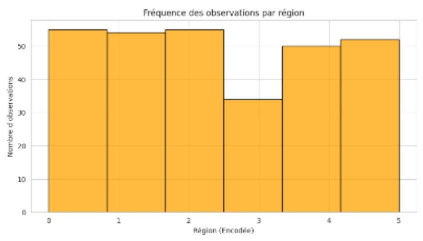


```python
df = pd.read_csv("prix_produits_agricoles_maroc_encoded.csv")

sns.set_style("whitegrid")



plt.figure(figsize=(10, 5))
sns.histplot(df["produit"], bins=len(df["produit"].unique()), color="green", edgecolor="black")
plt.title("Fréquence des produits agricoles")
plt.xlabel("Produit (Encodé)")
plt.ylabel("Nombre d'observations")
plt.show()

plt.figure(figsize=(10, 5))
sns.histplot(df["région"], bins=len(df["région"].unique()), color="orange", edgecolor="black")
plt.title("Fréquence des observations par région")
plt.xlabel("Région (Encodée)")
plt.ylabel("Nombre d'observations")
plt.show()
```

Histograms: Two histograms are created:



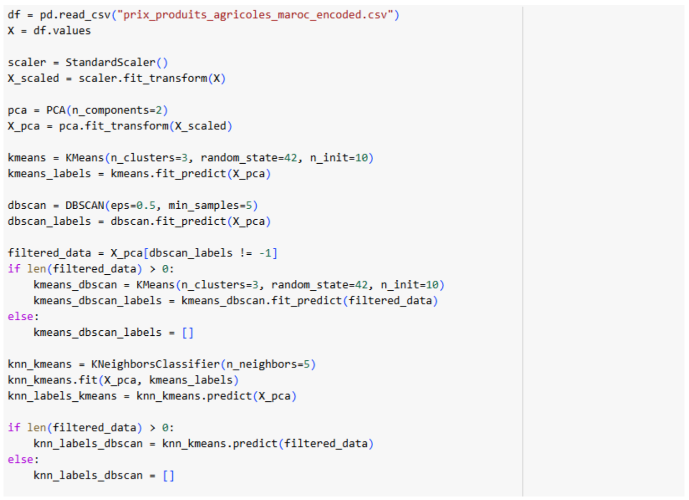
Product Frequency: Shows the frequency of each agricultural product in the dataset, providing insights into the relative popularity of each product.



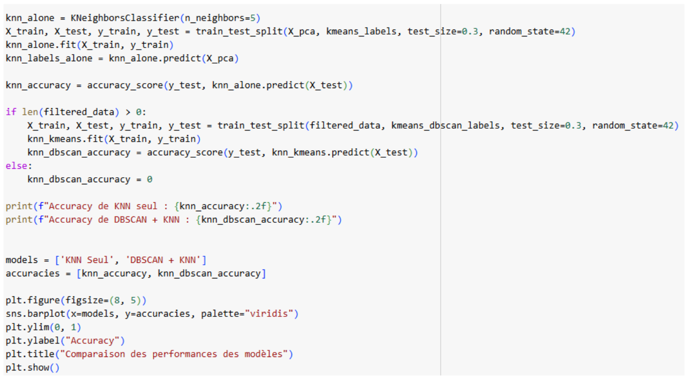
Region Frequency: Shows the frequency of observations by region, helping to identify any regional imbalances in the dataset.

3. Model Comparison

This script compares the performance of K-Nearest Neighbors (KNN) with and without DBSCAN



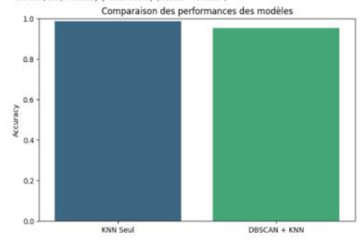
```python
df = pd.read_csv("prix_produits_agricoles_maroc_encoded.csv")
X = df.values

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

kmeans = KMeans(n_clusters=3, random_state=42, n_init=10)
kmeans_labels = kmeans.fit_predict(X_pca)

dbscan = DBSCAN(eps=0.5, min_samples=5)
dbscan_labels = dbscan.fit_predict(X_pca)

filtered_data = X_pca[dbscan_labels != -1]
if len(filtered_data) > 0:
    kmeans_dbscan = KMeans(n_clusters=3, random_state=42, n_init=10)
    kmeans_dbscan_labels = kmeans_dbscan.fit_predict(filtered_data)
else:
    kmeans_dbscan_labels = []

knn_kmeans = KNeighborsClassifier(n_neighbors=5)
knn_kmeans.fit(X_pca, kmeans_labels)
knn_labels_kmeans = knn_kmeans.predict(X_pca)

if len(filtered_data) > 0:
    knn_labels_dbscan = knn_kmeans.predict(filtered_data)
else:
    knn_labels_dbscan = []
```



```python
knn_alone = KNeighborsClassifier(n_neighbors=5)
X_train, X_test, y_train, y_test = train_test_split(X_pca, kmeans_labels, test_size=0.3, random_state=42)
knn_alone.fit(X_train, y_train)
knn_labels_alone = knn_alone.predict(X_pca)

knn_accuracy = accuracy_score(y_test, knn_alone.predict(X_test))

if len(filtered_data) > 0:
    X_train, X_test, y_train, y_test = train_test_split(filtered_data, kmeans_dbscan_labels, test_size=0.3, random_state=42)
    knn_kmeans.fit(X_train, y_train)
    knn_dbscan_accuracy = accuracy_score(y_test, knn_kmeans.predict(X_test))
else:
    knn_dbscan_accuracy = 0

print(f"Accuracy de KNN seul : {knn_accuracy:.2f}")
print(f"Accuracy de DBSCAN + KNN : {knn_dbscan_accuracy:.2f}")


models = ['KNN Seul', 'DBSCAN + KNN']
accuracies = [knn_accuracy, knn_dbscan_accuracy]

plt.figure(figsize=(8, 5))
sns.barplot(x=models, y=accuracies, palette="viridis")
plt.ylim(0, 1)
plt.ylabel("Accuracy")
plt.title("Comparaison des performances des modèles")
plt.show()
```

The analysis compares the performance of KNN with and without DBSCAN preprocessing. KNN alone achieved an accuracy of 0.99, while DBSCAN + KNN dropped to 0.96. The slight decrease suggests that DBSCAN might have removed useful information by filtering out points that were valuable for classification. The main idea is that DBSCAN's noise reduction didn't improve KNN accuracy in this case, possibly due to suboptimal parameters or the data being relatively clean. Further tuning and alternative noise reduction techniques are recommended for better results.

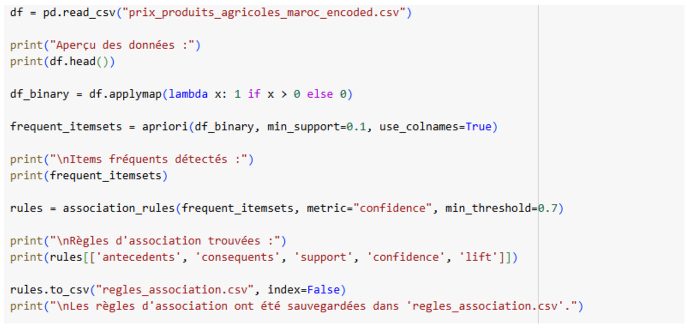
```
Accuracy de KNN seul : 0.99
Accuracy de DBSCAN + KNN : 0.96
<ipython-input-23-f62dbbd4dfd0>:76: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=models, y=accuracies, palette="viridis")
```



Based on this comparison, for this particular dataset and with the given settings, using KNN alone yields slightly better performance than using DBSCAN in combination with KNN. However, it's important to remember that this is just one comparison, and the results might vary with different datasets or parameter settings. Further experimentation and analysis are often necessary to draw more robust conclusions.

5 Apriori for Agricultural Price Analysis

This script aims to apply the **Apriori association rule mining algorithm** to discover interesting relationships within Moroccan agricultural product price data. The **Apriori algorithm** is used to identify frequent itemsets in the data and then generate association rules linking various attributes of the dataset. These rules help to understand the relationships between products, their prices, and other factors like region, year, and month. This process is executed using the **mlxtend library**.



```python
df = pd.read_csv("prix_produits_agricoles_maroc_encoded.csv")

print("Aperçu des données :")
print(df.head())

df_binary = df.applymap(lambda x: 1 if x > 0 else 0)

frequent_itemsets = apriori(df_binary, min_support=0.1, use_colnames=True)

print("\nItems fréquents détectés :")
print(frequent_itemsets)

rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.7)

print("\nRègles d'association trouvées :")
print(rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']])

rules.to_csv("regles_association.csv", index=False)
print("\nLes règles d'association ont été sauvegardées dans 'regles_association.csv'.")
```

- *Frequent Itemsets*

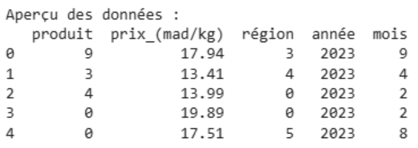
Frequent itemsets identify combinations of attributes that appear together regularly in the transactions. The **support** metric indicates the frequency at which these combinations occur. For example, an itemset where "product" appears with a support of 0.9 means that this product appears in 90% of the transactions, suggesting it's strongly represented in the data.

- *Association Rules*

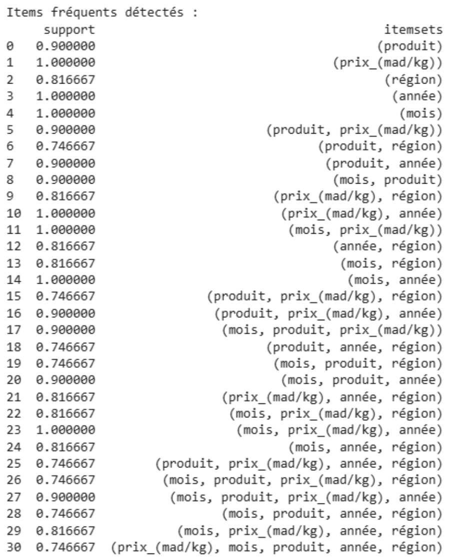
The generated rules in the form of "if A, then B" reveal relationships between the different attributes. For instance, a rule might state "if product X, then price Y", where **confidence** and **lift** measure the strength of the relationship:

- **Confidence**: Shows the probability of B occurring when A occurs. A higher confidence (greater than 70%) indicates a strong relationship between A and B.
- **Lift**: Measures the increase in the likelihood of B occurring when A is present, compared to what would be expected by chance. A lift greater than 1 indicates a positive correlation between A and B, while a lift less than 1 suggests that the two attributes may be independent.
- *Interpretation of Results*

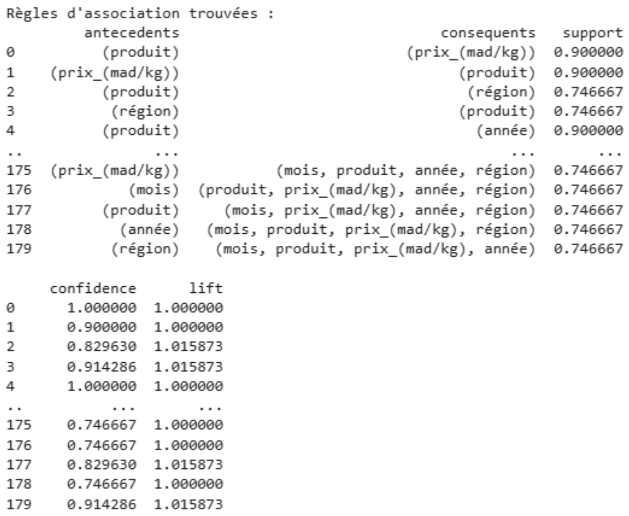
The results highlight potential associations between agricultural product prices and other attributes. For example, a rule with high confidence and a lift greater than 1 suggests that the combination of the two attributes is more significant than expected. Frequent itemsets also help identify attribute groupings that often appear together in the data.



```
Aperçu des données :
   produit  prix_(mad/kg)  région  année  mois
0        9          17.94       3   2023     9
1        3          13.41       4   2023     4
2        4          13.99       0   2023     2
3        0          19.89       0   2023     2
4        0          17.51       5   2023     8
```



```
Items fréquents détectés :
     support                                     itemsets
0   0.900000                                    (produit)
1   1.000000                              (prix_(mad/kg))
2   0.816667                                     (région)
3   1.000000                                      (année)
4   1.000000                                       (mois)
5   0.900000                     (produit, prix_(mad/kg))
6   0.746667                            (produit, région)
7   0.900000                             (produit, année)
8   0.900000                              (mois, produit)
9   0.816667                      (prix_(mad/kg), région)
10  1.000000                       (prix_(mad/kg), année)
11  1.000000                        (mois, prix_(mad/kg))
12  0.816667                              (année, région)
13  0.816667                               (mois, région)
14  1.000000                                (mois, année)
15  0.746667             (produit, prix_(mad/kg), région)
16  0.900000              (produit, prix_(mad/kg), année)
17  0.900000               (mois, produit, prix_(mad/kg))
18  0.746667                     (produit, année, région)
19  0.746667                      (mois, produit, région)
20  0.900000                       (mois, produit, année)
21  0.816667               (prix_(mad/kg), année, région)
22  0.816667                (mois, prix_(mad/kg), région)
23  1.000000                 (mois, prix_(mad/kg), année)
24  0.816667                        (mois, année, région)
25  0.746667      (produit, prix_(mad/kg), année, région)
26  0.746667       (mois, produit, prix_(mad/kg), région)
27  0.900000        (mois, produit, prix_(mad/kg), année)
28  0.746667               (mois, produit, année, région)
29  0.816667         (mois, prix_(mad/kg), année, région)
30  0.746667  (prix_(mad/kg), mois, produit, année, région)
```



```
Règles d'association trouvées :
         antecedents                             consequents   support
0          (produit)                         (prix_(mad/kg))  0.900000
1    (prix_(mad/kg))                               (produit)  0.900000
2          (produit)                                (région)  0.746667
3           (région)                               (produit)  0.746667
4          (produit)                                 (année)  0.900000
..               ...                                     ...       ...
175  (prix_(mad/kg))            (mois, produit, année, région)  0.746667
176           (mois)  (produit, prix_(mad/kg), année, région)  0.746667
177        (produit)     (mois, prix_(mad/kg), année, région)  0.746667
178          (année)   (mois, produit, prix_(mad/kg), région)  0.746667
179         (région)    (mois, produit, prix_(mad/kg), année)  0.746667

     confidence      lift
0      1.000000  1.000000
1      0.900000  1.000000
2      0.829630  1.015873
3      0.914286  1.015873
4      1.000000  1.000000
..          ...       ...
175    0.746667  1.000000
176    0.746667  1.000000
177    0.829630  1.015873
178    0.746667  1.000000
179    0.914286  1.015873
```

6 Conclusion

This project applies various machine learning techniques to analyze agricultural product price data in Morocco. It incorporates **K-Means**, **DBSCAN**, and **K-Nearest Neighbors (KNN)** for clustering and classification, using **Principal Component Analysis (PCA)** for dimensionality reduction. The goal was to compare the performance of KNN alone and KNN combined with DBSCAN, finding that DBSCAN slightly reduced the model's accuracy in this case. Additionally, the **Apriori algorithm** was used for association rule mining, uncovering frequent itemsets and generating rules that revealed strong relationships between attributes, offering valuable insights for agricultural stakeholders. Overall, the project demonstrated how clustering, classification, and association mining can provide actionable insights from the data.

7 References

Aitelmouden, K. (2025). *Prix des produits agricoles au Maroc*. Kaggle. Retrieved from https://www.kaggle.com/datasets/khaoulaaitelmouden/prix-des-produits-agricoles-au-maroc.

Google. (2025). *Google Colab*. Retrieved from https://colab.research.google.com/drive/14AP0XSuzkHdhg3csJ4HyXJby9GlOTmMf?usp=sharing.
[Comment: This link includes all my code in Google Colab.]

---

8 Tool Acknowledgment

The analysis was conducted using **Google Colab**, an interactive web-based tool provided by Google that allows for running Python code in a cloud environment with ease, facilitating collaboration and ease of access. The following Colab notebook was used for the analysis: *Google Colab Notebook*.

Python (Python Software Foundation, 2025) was the primary programming language used for implementing the algorithms and data analysis in this work.