

Compte rendu

Apprentissage supervisé et non supervisé

Réalisée par :

Ihssane BAMMAD

Sommaire :

I. Introduction.....	3
II. Problématique :	3
1. Définition de l'ACP	3
2. Rôle de l'ACP	4
III. Analyse en Composantes Principales des Performances des Athlètes.....	4
1. Chargement des données	4
2. Explication des données	5
3. Standardisation des données pour l'ACP	7
4. Réalisation de l'ACP et justification	8
4.1. Utilité des variables.....	9
4.3. Visualisation des résultats	11
5. Représentation des Valeurs Propres	12
6. Nombre d'axes conservées	13
6.1. Analyse du Graphe de la Variance Cumulée :	13
6.2. Valeurs Propres :	14
7. Représentation Graphique des Athlètes et des Sports	14
7.1. Interprétation Globale :	15
7.2. Profils des Athlètes :	16
IV. Conclusion de l'Exercice	19

I. Introduction

Dans le cadre de cette analyse, nous nous intéressons aux performances des athlètes de décathlon lors des Jeux Olympiques de 2004 et de l'événement Décastar 2004. Le décathlon est une discipline combinée qui évalue les athlètes sur 10 épreuves distinctes, englobant des compétences variées telles que la vitesse, la force, l'endurance et la technique. Chaque athlète accumule des points en fonction de ses performances dans ces épreuves, aboutissant à un classement final.

L'objectif de cette étude est de réaliser une analyse en composantes principales (ACP) afin d'explorer les performances des athlètes. L'ACP est une technique statistique qui permet de réduire la dimensionnalité d'un jeu de données tout en conservant l'essentiel de la variance. Cette méthode est particulièrement utile dans ce contexte, où les performances des athlètes sont évaluées sur un grand nombre de variables. En projetant ces variables dans un espace de dimension réduite, nous espérons extraire des informations pertinentes qui permettront d'explorer des corrélations cachées et de définir des profils typiques d'athlètes.

II. Problématique :

L'enjeu de cette étude est de comprendre les relations entre les performances des athlètes dans les différentes épreuves du décathlon et d'identifier les principales caractéristiques qui influencent leur performance globale. En utilisant l'analyse en composantes principales (ACP), nous visons à réduire la complexité des données tout en conservant l'essentiel de la variance. Cette réduction de la dimension permettra d'obtenir une vue d'ensemble des performances et de dégager des profils types d'athlètes en fonction de leurs résultats. Il s'agit de mettre en évidence les corrélations sous-jacentes entre les épreuves et de faciliter l'interprétation des performances athlétiques dans un espace réduit.

1. Définition de l'ACP

L'Analyse en Composantes Principales (ACP) est une méthode statistique de réduction de la dimensionnalité qui permet de transformer un ensemble de variables corrélées en un ensemble

de variables nouvelles non corrélées, appelées composantes principales. Chaque composante principale est une combinaison linéaire des variables initiales, et les premières composantes principales capturent la majeure partie de la variance des données.

2. Rôle de l'ACP

L'objectif principal de l'ACP est de simplifier les données en réduisant le nombre de dimensions, tout en conservant le maximum d'information possible. Elle permet de visualiser des données complexes dans un espace de plus faible dimension, de détecter des structures cachées, et de faciliter l'interprétation en identifiant les axes de variations les plus importants. Cette approche est particulièrement utile pour l'exploration des données, la compression d'information et la préparation des données pour des algorithmes d'apprentissage automatique.


III. Analyse en Composantes Principales des Performances des Athlètes

Cette analyse vise à explorer les performances des athlètes ayant participé aux épreuves de décathlon lors des Jeux Olympiques 2004 et du Décastar 2004. Grâce à l'Analyse en Composantes Principales (ACP), nous réduirons la dimensionnalité des données tout en conservant un maximum d'information. Cette méthode nous permettra d'identifier des tendances, de visualiser les relations entre les différentes épreuves et de dégager des profils types d'athlètes, tout en facilitant la classification des individus selon leurs performances globales.

1. Chargement des données

Dans cette première étape, nous allons charger le jeu de données "Décathlon" disponible sur Kaggle, qui contient les performances des athlètes lors des compétitions JO 2004 et Décastar 2004.

Le code ci-dessous utilise **Google Colab** pour accéder aux données stockées sur Google Drive. Nous montons d'abord Google Drive, puis nous utilisons la bibliothèque **pandas** pour lire le fichier CSV contenant les données.

```
✓ 30s  from google.colab import drive
import pandas as pd
drive.mount('/content/drive/')
fch=pd.read_csv('/content/drive/My Drive/decaathlon.csv')
|


 Mounted at /content/drive/
```

Figure :Code de téléchargement de la data

2. Explication des données

Le jeu de données "Décathlon" contient les performances des athlètes dans différentes épreuves du décathlon lors des compétitions JO 2004 et Décastar 2004. Chaque ligne représente un athlète, et chaque colonne fournit des informations sur ses performances dans une ou plusieurs disciplines. Voici la description des colonnes présentes dans les données :

- **Athlets** : Le nom de l'athlète qui participe à la compétition.
- **100m** : Temps réalisé (en secondes) par l'athlète lors de l'épreuve du 100 mètres, une course de sprint.
- **Long.jump** : Distance atteinte (en mètres) lors de l'épreuve du saut en longueur.
- **Shot.put** : Distance atteinte (en mètres) lors de l'épreuve du lancer du poids.
- **High.jump** : Hauteur franchie (en mètres) lors de l'épreuve du saut en hauteur.
- **400m** : Temps réalisé (en secondes) lors de l'épreuve du 400 mètres, une course d'endurance rapide.
- **110m.hurdle** : Temps réalisé (en secondes) lors de l'épreuve du 110 mètres haies, une course comportant des obstacles.
- **Discus** : Distance atteinte (en mètres) lors de l'épreuve du lancer du disque.
- **Pole.vault** : Hauteur franchie (en mètres) lors de l'épreuve du saut à la perche.
- **Javeline** : Distance atteinte (en mètres) lors de l'épreuve du lancer de javelot.
- **1500m** : Temps réalisé (en minutes et secondes) lors de l'épreuve du 1500 mètres, une course de demi-fond.
- **Rank** : Le classement final de l'athlète à l'issue des 10 épreuves du décathlon.

- **Points** : Le nombre total de points accumulés par l'athlète en fonction de ses performances dans les 10 épreuves. Chaque épreuve a un système de points spécifique basé sur la performance.
- **Competition** : Le type de compétition à laquelle l'athlète a participé, soit les Jeux Olympiques 2004 (JO 2004) soit Décastar 2004.

```
# 2. Explication des données
print(fch.columns.tolist()) # Affiche les noms des colonnes
print("Premières lignes du dataset :")
print(fch.head())

print("\nInformations sur les données :")
print(fch.info())

print("\nDescription statistique des données :")
print(fch.describe())
```

Figure :code de visualisation de la data

Ce bloc de code permet d'explorer rapidement le dataset "Décathlon". La commande `fch.columns.tolist()` affiche les colonnes, offrant une vue d'ensemble des variables. Ensuite, `fch.head()` montre les cinq premières lignes, fournissant un aperçu des données. La fonction `fch.info()` donne des informations sur le nombre d'entrées, le type de données et les valeurs non nulles, tandis que `fch.describe()` fournit une description statistique des colonnes numériques, incluant la moyenne, l'écart type et les valeurs minimales et maximales. En résumé, ce code aide à comprendre la structure et la distribution des performances des athlètes, essentielle pour la standardisation et l'analyse en composantes principales (ACP).

Nous obtenons comme résultat :

```
[ 'Athlets', '100m', 'Long.jump', 'Shot.put', 'High.jump', '400m', '110m.hurdle', 'Discus', 'Pole.vault', 'Javeline', '1500m', 'Rank', 'Points', 'Competition' ]
```

Premières lignes du dataset :

	Athlets	100m	Long.jump	Shot.put	High.jump	400m	110m.hurdle	Discus	
0	SEBRLE	11.04	7.58	14.83	2.07	49.81	14.69	43.75	
1	CLAY	10.76	7.40	14.26	1.86	49.37	14.05	50.72	
2	KARPOV	11.02	7.30	14.77	2.04	48.37	14.09	48.95	
3	BERNARD	11.02	7.23	14.25	1.92	48.93	14.99	40.87	
4	YURKOV	11.34	7.09	15.19	2.10	50.42	15.31	46.26	

	Pole.vault	Javeline	1500m	Rank	Points	Competition
0	5.02	63.19	291.7	1	8217	Decastar
1	4.92	60.15	301.5	2	8122	Decastar
2	4.92	50.31	300.2	3	8099	Decastar
3	5.32	62.77	280.1	4	8067	Decastar
4	4.72	63.44	276.4	5	8036	Decastar

Figure : visualisation du contenu de la data

```

➡ Informations sur les données :
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41 entries, 0 to 40
Data columns (total 14 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Athlets               41 non-null    object
1   100m                  41 non-null    float64
2   Long.jump             41 non-null    float64
3   Shot.put              41 non-null    float64
4   High.jump             41 non-null    float64
5   400m                  41 non-null    float64
6   110m.hurdle           41 non-null    float64
7   Discus                41 non-null    float64
8   Pole.vault            41 non-null    float64
9   Javeline              41 non-null    float64
10  1500m                 41 non-null    float64
11  Rank                  41 non-null    int64
12  Points                41 non-null    int64
13  Competition           41 non-null    object
dtypes: float64(10), int64(2), object(2)
memory usage: 4.6+ KB
None

```

Figure : Information sur les types des données

```

➡ Description statistique des données :

```

	100m	Long.jump	Shot.put	High.jump	400m	110m.hurdle
count	41.000000	41.000000	41.000000	41.000000	41.000000	41.000000
mean	10.998049	7.260000	14.477073	1.976829	49.616341	14.605854
std	0.263023	0.316402	0.824428	0.088951	1.153451	0.471789
min	10.440000	6.610000	12.680000	1.850000	46.810000	13.970000
25%	10.850000	7.030000	13.880000	1.920000	48.930000	14.210000
50%	10.980000	7.300000	14.570000	1.950000	49.400000	14.480000
75%	11.140000	7.480000	14.970000	2.040000	50.300000	14.980000
max	11.640000	7.960000	16.360000	2.150000	53.200000	15.670000

	Discus	Pole.vault	Javeline	1500m	Rank	Points
count	41.000000	41.000000	41.000000	41.000000	41.000000	41.000000
mean	44.325610	4.762439	58.316585	279.024878	12.121951	8005.365854
std	3.377845	0.278000	4.826820	11.673247	7.918949	342.385145
min	37.920000	4.200000	50.310000	262.100000	1.000000	7313.000000
25%	41.900000	4.500000	55.270000	271.020000	6.000000	7802.000000
50%	44.410000	4.800000	58.360000	278.050000	11.000000	8021.000000
75%	46.070000	4.920000	60.890000	285.100000	18.000000	8122.000000
max	51.650000	5.400000	70.520000	317.000000	28.000000	8893.000000

Figure :la description statistique de la data

3. Standardisation des données pour l'ACP

Avant de réaliser une analyse en composantes principales (ACP), il est essentiel de centrer et de normaliser les variables quantitatives. Cette étape, connue sous le nom de standardisation, vise à rendre les données comparables, surtout lorsque les variables sont mesurées sur des échelles différentes. En centrant les données, on soustrait la moyenne de chaque variable, ce qui permet d'obtenir une distribution centrée autour de zéro. Ensuite, en normalisant, on divise chaque valeur par l'écart type, ce qui garantit que chaque variable contribue de manière équitable à l'analyse, en ayant une variance similaire.

```
import numpy as np
from sklearn.preprocessing import StandardScaler
# Sélectionner les colonnes numériques pour l'ACP
features = fch.select_dtypes(include=[np.number]).columns
X = fch[features].values

# 3. Sélectionner les colonnes correspondant aux performances (10 colonnes)
performance_columns = fch.columns[1:11]

# 4. Standardisation des données
scaler = StandardScaler()
X_scaled = scaler.fit_transform(fch[performance_columns])
```

Dans le code ci-dessus, les colonnes numériques sont sélectionnées à partir du DataFrame contenant les performances des athlètes. La méthode `StandardScaler` de la bibliothèque `sklearn.preprocessing` est utilisée pour effectuer la standardisation. D'abord, les colonnes pertinentes pour l'ACP sont identifiées, puis les données de performance sont transformées à l'aide de `fit_transform`, produisant ainsi un nouvel ensemble de données (`X_scaled`) où chaque variable a une moyenne de zéro et un écart type de un. Cette standardisation permet de s'assurer que l'ACP peut capturer correctement les relations entre les variables sans être biaisée par des différences d'échelle.

4. Réalisation de l'ACP et justification

L'analyse en composantes principales (ACP) est réalisée sur les dix colonnes du jeu de données qui correspondent aux performances des athlètes dans les épreuves du décathlon. Cette méthode est utilisée pour réduire la dimensionnalité des données tout en préservant autant que possible la variance présente dans celles-ci. En réduisant le nombre de dimensions, l'ACP permet d'identifier des motifs et des relations sous-jacents dans les données, facilitant ainsi la visualisation et l'interprétation des résultats.

Dans ce cas, l'ACP est particulièrement utile car les performances des athlètes dans les différentes épreuves du décathlon sont souvent corrélées. Par exemple, un athlète performant au 100 m peut également exceller dans le saut en longueur ou le lancer du poids. En regroupant ces performances en composantes principales, nous pouvons simplifier l'analyse tout en capturant l'essentiel de la variation des données.

4.1. Utilité des variables

- **Rank** : Indique la position finale d'un athlète, essentielle pour évaluer la performance relative et établir des corrélations avec les résultats de l'ACP. Cela permet d'identifier les athlètes ayant des performances distinctes.
- **Points** : Reflète l'efficacité globale des performances. Cette variable aide à repérer les athlètes constants, même ceux en dehors du top 3, ce qui est utile pour les décisions des entraîneurs.
- **Compétition** : Fournit le contexte des événements, influençant les performances par des facteurs comme la météo et la concurrence. Son inclusion permet d'analyser les variations de performances selon les compétitions.

4.2. Réalisation de PCA

```
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
# 5. Réaliser l'ACP
pca = PCA(n_components=10) # Nous choisissons de réduire à 2 dimensions
X_pca = pca.fit_transform(X_scaled) # Transforme les données

# 6. Visualiser les résultats de l'ACP
plt.figure(figsize=(8, 5))
plt.scatter(X_pca[:, 0], X_pca[:, 1], alpha=0.7)
plt.title('Analyse en Composantes Principales (ACP)')
plt.xlabel('Première Composante Principale')
plt.ylabel('Deuxième Composante Principale')
plt.grid()
plt.show()

# 7. Variance expliquée par chaque composante
explained_variance = pca.explained_variance_ratio_
print("Variance expliquée par chaque composante :", explained_variance)
print("Variance cumulée :", np.cumsum(explained_variance))
# 5. Inclure les variables 'rank', 'points', et 'competition' pour analyse ultérieure
fch_selected = fch[performance_columns.tolist() + ['Rank', 'Points', 'Competition']]
```

Figure :code de PCA

Le code réalise une Analyse en Composantes Principales (ACP) ,Donc après avoir importé les bibliothèques nécessaires, le code initialise le modèle PCA pour réduire les données à 10 dimensions et les transforme en utilisant la méthode `fit_transform`. Les résultats sont visualisés à l'aide d'un graphique de dispersion. Enfin, le code calcule et affiche la variance expliquée par chaque composante ainsi que la variance cumulée. Les variables `Rank`, `Points`,

et Compétition sont également sélectionnées pour une analyse ultérieure, enrichissant l'interprétation des performances athlétiques. ce code nous donne comme résultat :

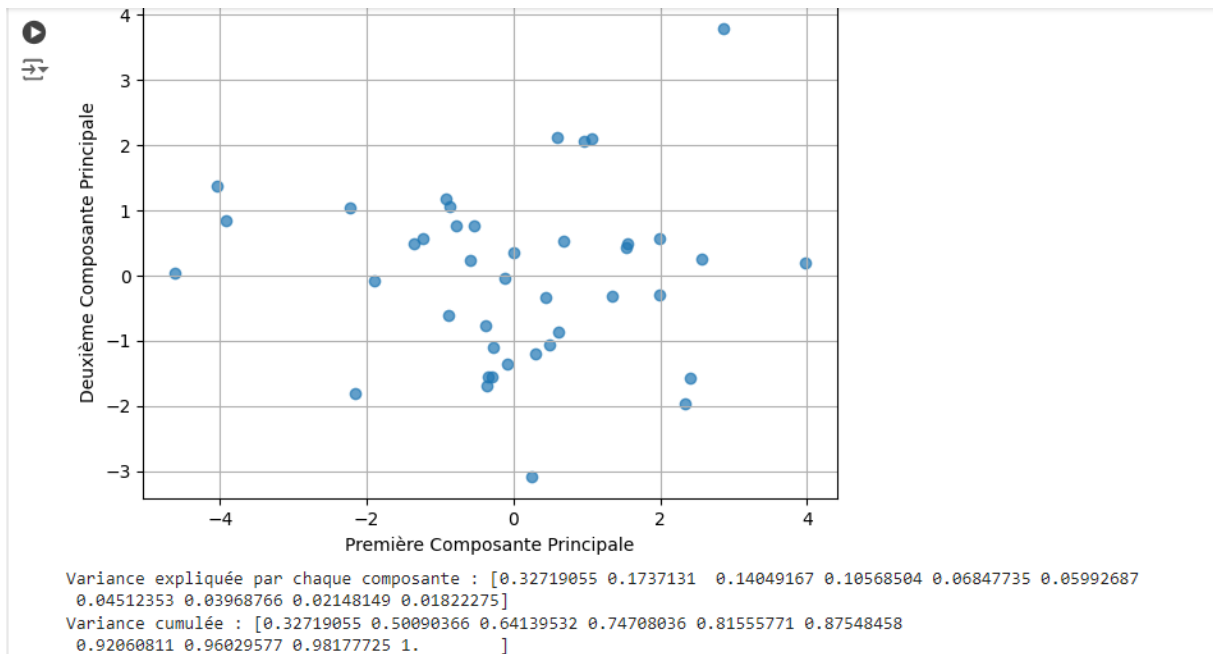


Figure : Résultat de PCA

- Les résultats de l'Analyse en Composantes Principales (ACP) montrent la variance expliquée par chaque composante principale, avec la première composante (0.327) expliquant plus d'un tiers de la variance totale. La variance cumulée indique que les deux premières composantes combinées représentent environ 50% de la variance totale des performances des athlètes.

Le graphique de dispersion illustre les athlètes dans un espace réduit défini par les deux premières composantes principales, permettant d'identifier des regroupements ou des clusters basés sur leurs performances. Cela peut aider à visualiser des tendances, comme la séparation entre les athlètes de haut niveau et ceux de niveau inférieur.

4.3. Visualisation des résultats

Après avoir réalisé l'ACP et obtenu les résultats de variance expliquée pour chaque composante principale, il est utile de visualiser ces résultats à l'aide de diagrammes à barres pour mieux comprendre la contribution de chaque composante à la variance totale.

Le code ci-dessous calcule la variance cumulée et représente graphiquement la variance expliquée par chaque composante ainsi que la variance cumulée. Cela nous permet de voir combien de composantes sont nécessaires pour expliquer une grande partie de la variance. En ajoutant une ligne pour indiquer le seuil de 95 %, nous pouvons déterminer le nombre optimal de dimensions à retenir.

```
cumulative_variance = np.cumsum(explained_variance)
# Conversion en pourcentage
explained_variance_percentage = explained_variance * 100
cumulative_variance_percentage = cumulative_variance * 100
# Création du graphique
plt.figure(figsize=(10, 4))
# Graphique de la variance expliquée par chaque composante
plt.subplot(1, 2, 1)
sns.barplot(x=np.arange(1, len(explained_variance) + 1), y=explained_variance_percentage, palette='viridis')
plt.title('Variance Expliquée par Composante Principale (%)')
plt.xlabel('Composantes Principales')
plt.ylabel('Variance Expliquée (%)')
plt.xticks(ticks=np.arange(1, len(explained_variance) + 1), labels=np.arange(1, len(explained_variance) + 1))
# Graphique de la variance cumulée
plt.subplot(1, 2, 2)
plt.plot(np.arange(1, len(cumulative_variance) + 1), cumulative_variance_percentage, marker='o', color='b')
plt.title('Variance Cumulée des Composantes Principales (%)')
plt.xlabel('Composantes Principales')
plt.ylabel('Variance Cumulée (%)')
plt.xticks(ticks=np.arange(1, len(cumulative_variance) + 1), labels=np.arange(1, len(cumulative_variance) + 1))
plt.axhline(y=95, color='r', linestyle='--') # Ligne pour la barre de 95%
plt.text(1, 95, '95%', color='red', verticalalignment='bottom')
plt.tight_layout()
plt.show()
```

Figure : Code de visualisation de la variance

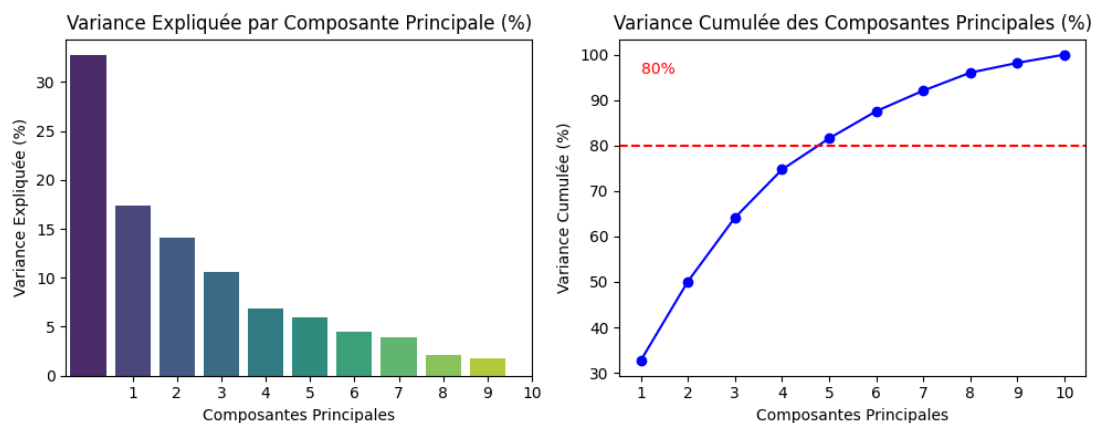


Figure : graphe de variance

- Dans le premier graphique, nous observons 10 barres représentant la variance expliquée par chaque composante principale. La première barre, correspondant à la première composante principale, affiche une variance expliquée d'environ 32 %, ce qui indique qu'elle capture une part significative des informations dans les données. Les barres suivantes montrent une diminution progressive, culminant à environ 1,8 % pour la dixième composante. Cela signifie que les premières composantes principales sont beaucoup plus informatives que les dernières, suggérant qu'une grande partie de la variance des données est concentrée dans quelques dimensions.

Dans le deuxième graphique, la variance cumulée est représentée par des points sur un axe croissant. Nous remarquons que la courbe atteint le seuil de 80 % de variance expliquée dès la cinquième composante principale. Cela indique que l'ajout des cinq premières composantes permet d'expliquer presque toutes les variations présentes dans les données. Par la suite, la courbe commence à se stabiliser, ce qui signifie que les composantes restantes n'apportent que peu ou pas d'informations supplémentaires.

5. Représentation des Valeurs Propres

Le code ci-dessus trace les valeurs propres associées aux composantes principales de l'Analyse en Composantes Principales (ACP). Les valeurs propres représentent la quantité de variance capturée par chaque composante principale. Dans ce graphique, chaque barre correspond à une composante principale et sa hauteur indique la valeur propre associée.

```
# Valeurs propres
eigenvalues = pca.explained_variance_
# Tracer les valeurs propres
plt.figure(figsize=(10, 4))
plt.bar(range(1, len(eigenvalues) + 1), eigenvalues, alpha=0.7)
plt.title('Valeurs Propres des Composantes Principales')
plt.xlabel('Composantes Principales')
plt.ylabel('Valeurs Propres')
plt.xticks(range(1, len(eigenvalues) + 1)) # Labels for the x-axis
plt.grid()
plt.show()
```

Figure : code des valeurs propres.

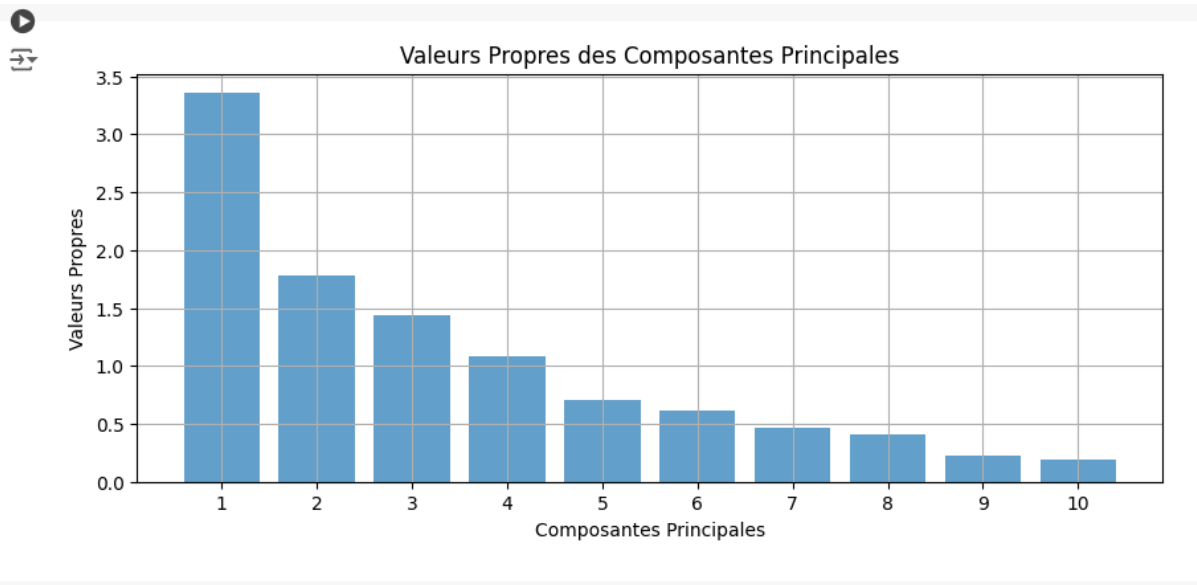


Figure : graphes des valeurs propres des PC

- Dans le graphe ci-dessus, chaque valeur propre correspond à la quantité de variance capturée par une composante principale. Une valeur propre plus élevée indique que la composante capture une plus grande partie de la variance des données, ce qui signifie qu'elle est plus informative.

6. Nombre d'axes conservées

Pour déterminer le nombre d'axes à conserver pour un processus d'apprentissage tel que K-means, nous pouvons nous baser sur l'analyse de la variance cumulée et des valeurs propres.

6.1. Analyse du Graphe de la Variance Cumulée :

Dans le graphe de la variance cumulée, nous observons que le point 5 atteint un seuil de plus de 80% de variance expliquée. Cela signifie qu'en conservant seulement 5 axes, nous capturons une proportion significative des informations présentes dans les données, ce qui est crucial pour le processus de clustering. En effet, K-means tire parti de cette information pour identifier des clusters pertinents.

6.2. Valeurs Propres :

En consultant les valeurs propres des composantes principales, on remarque que les premières composantes ont des valeurs significativement plus élevées par rapport aux suivantes. Cela confirme que les premières composantes (1 à 5) contiennent l'essentiel de la variance, tandis que les composantes suivantes contribuent marginalement à l'explication de la variance. Par conséquent, conserver ces 5 premiers axes est judicieux pour maximiser la qualité des clusters tout en minimisant le bruit.

- Enfin, conserver 5 axes représente un bon compromis, car ces axes permettent de capturer 81.56% de la variance totale. Cela réduit la complexité du modèle et facilite l'interprétation des résultats du clustering tout en garantissant que nous ne perdons pas d'informations critiques.

7. Représentation Graphique des Athlètes et des Sports

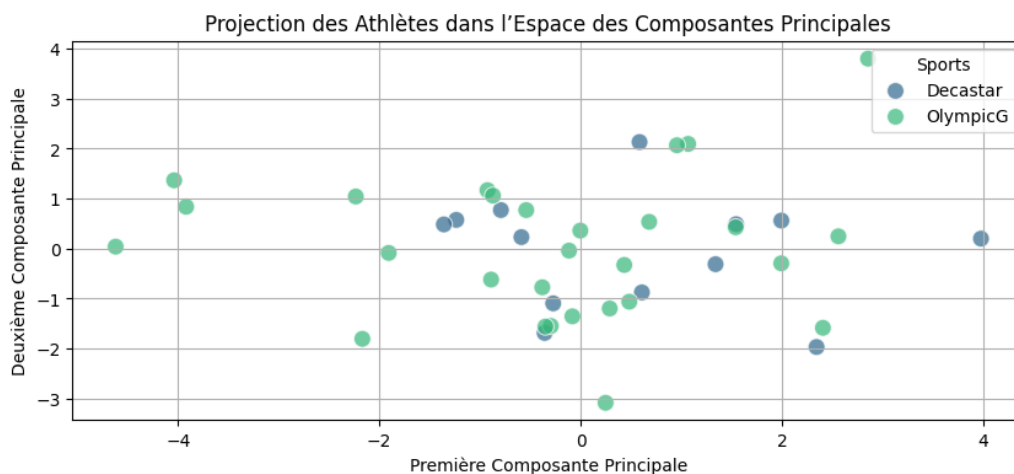
Pour répondre à la question sur la représentation graphique des individus (athlètes) et des sports, nous allons visualiser les données projetées sur les deux premières composantes principales. Le code ci-dessous génère un graphique de dispersion qui illustre cette projection

```
import seaborn as sns

# Supposons que 'Compétition' est bien la colonne qui représente les sports
sports = fch_selected['Compétition'] # Extraire la colonne des sports

# Créer le graphique de dispersion
plt.figure(figsize=(10, 4))
sns.scatterplot(x=X_pca[:, 0], y=X_pca[:, 1], hue=sports, palette='viridis', s=100, alpha=0.7)

plt.title('Projection des Athlètes dans l'Espace des Composantes Principales')
plt.xlabel('Première Composante Principale')
plt.ylabel('Deuxième Composante Principale')
plt.grid()
plt.legend(title='Sports')
plt.show()
```



7.1. Interprétation Globale :

Athlètes Olympiques :

La majorité des athlètes Olympiques présentent une forte performance dans certaines dimensions clés, ce qui indique qu'ils possèdent des compétences spécifiques qui les distinguent dans leur discipline. Cependant, un nombre significatif d'entre eux montre également des performances faibles dans d'autres domaines, suggérant des lacunes qui pourraient être améliorées.

Cela indique que bien que certains athlètes soient très performants, il existe un potentiel d'optimisation pour renforcer les aspects de leur entraînement qui pourraient les aider à atteindre un niveau d'excellence encore plus élevé.

Athlètes Décastar :

Les athlètes Décastar montrent une répartition plus variée de leurs performances, ce qui reflète la nature multidisciplinaire de ce sport. Ils semblent avoir un équilibre de compétences, bien que certains puissent également avoir des zones à développer.

La présence d'athlètes à la fois performants et moins performants souligne la nécessité d'un entraînement ciblé pour maximiser leur potentiel global dans les différentes épreuves.

7.2. Profils des Athlètes :

Oui, il est tout à fait possible de définir des profils d'athlètes en fonction des résultats de l'analyse des composantes principales. Voici quelques profils possibles basés sur la distribution des athlètes des sports Décastar et Olympique dans l'espace des performances :

➤ Profil Olympique Hautement Performant :

- Ce groupe comprend des athlètes Olympiques qui affichent des performances élevées dans les dimensions mesurées, se distinguant par des compétences spécifiques. Ces athlètes peuvent exceller dans les épreuves techniques ou tactiques, leur permettant de se classer au sommet de leur discipline.

➤ Profil Olympique avec Potentiel d'Amélioration :

- Ce profil inclut des athlètes qui, bien qu'ils soient classés comme Olympiques, présentent des performances mitigées. Ils ont le potentiel d'améliorer certaines compétences ou dimensions de performance pour atteindre un niveau supérieur. Ils peuvent bénéficier d'une formation ciblée pour développer leurs faiblesses.

➤ Profil Décastar Multidisciplinaire :

- Les athlètes Décastar se caractérisent par une diversité de compétences, montrant une certaine uniformité dans leurs performances à travers plusieurs disciplines. Ce profil indique une bonne capacité d'adaptation et de polyvalence, ce qui est crucial dans un sport qui exige de la performance dans différentes épreuves.

➤ Profil Décastar avec Zones d'Amélioration :

8. Ce groupe comprend des athlètes Décastar qui ont des performances inégales, avec des points forts dans certaines épreuves mais des faiblesses dans d'autres. Profils Gagnants des Athlètes Basés sur les Compétitions

Analyse des Profils Gagnants

Pour répondre à cette question on est besoin de visualiser les performances des athlètes dans les compétitions en fonction de leur rang et points :

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

def define_profile(row):
    if row['Rank'] == 1:
        return 'Profil Olympique Hautement Performant'
    elif row['Points'] > 800:
        return 'Profil Olympique avec Potentiel d\'Amélioration'
    elif row['Competition'] == 'Decastar':
        return 'Profil Décastar Multidisciplinaire'
    else:
        return 'Profil Décastar avec Zones d\'Amélioration'

# Appliquer la fonction à chaque ligne pour créer une nouvelle colonne 'Profile'
fch_selected['Profile'] = fch_selected.apply(define_profile, axis=1)
# Visualisation
plt.figure(figsize=(12, 6))
sns.scatterplot(data=fch_selected, x='Points', y='Rank', hue='Competition', style='Profile', s=100)

plt.title('Visualisation des Profils Gagnants par Compétition')
plt.xlabel('Points')
plt.ylabel('Rang')
plt.axhline(y=1, color='r', linestyle='--') # Ligne pour indiquer le premier rang
plt.grid()
plt.legend(title='Compétition')
plt.show()
```

Figure : Code d'analyse des gagnants

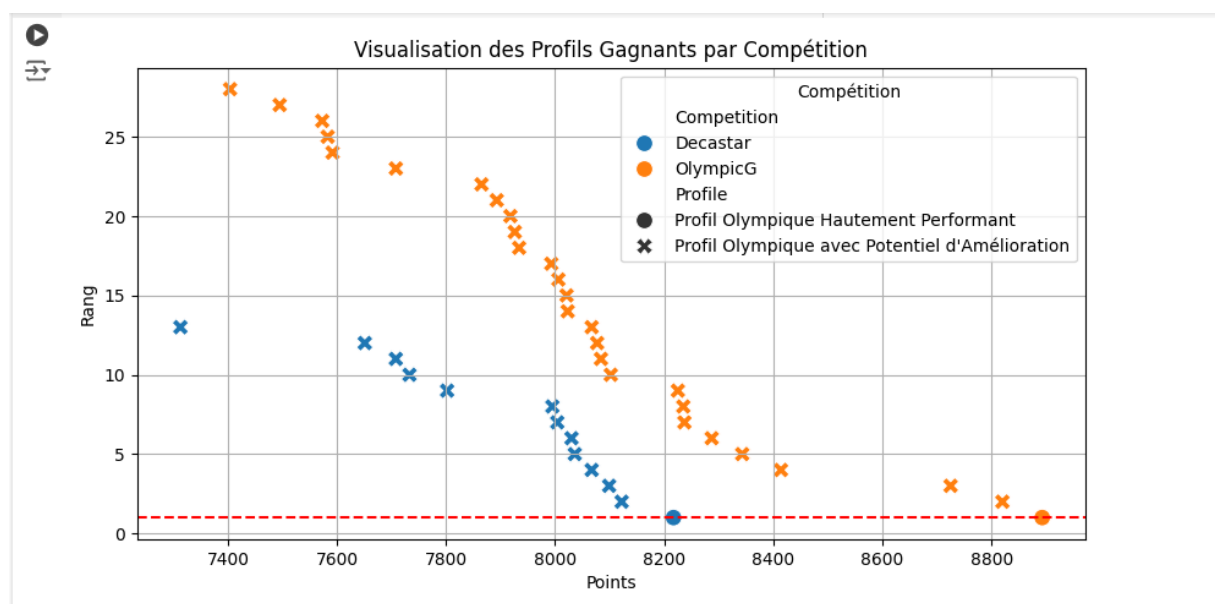


Figure : graphe de visualisation des profils gagnants.

À partir des données et des visualisations trouvées, nous avons pu identifier plusieurs profils d'athlètes qui se distinguent dans les compétitions, en particulier dans les catégories Olympique et Décastar.

1. Profil Olympique Hautement Performant :

Ces athlètes se caractérisent par des scores de points très élevés et un rang inférieur à 3. Ils dominent les compétitions en affichant des performances exceptionnelles qui les placent en tête de classement.

2. Profil Décastar Hautement Performant :

Ce profil inclut des athlètes ayant également des rangs en dessous de 3, mais avec des scores de points généralement plus bas que ceux des athlètes olympiques. Ils affichent une grande polyvalence dans plusieurs disciplines, ce qui leur permet d'obtenir de bonnes performances, même si leurs scores sont légèrement inférieurs à ceux des olympiques.

3. Profil Olympique avec Potentiel d'Amélioration :

Certains athlètes dans la catégorie olympique montrent des scores de points intermédiaires mais possèdent des rangs encore compétitifs. Ces athlètes ont un potentiel d'amélioration significatif et peuvent bénéficier d'un entraînement ciblé pour atteindre des niveaux de performance encore plus élevés.

4. Profil Décastar avec Zones d'Amélioration :

Ce profil regroupe des athlètes dont les scores de points sont plus bas) et des rangs plus élevés (au-dessus de 10). Ils montrent des signes de compétitivité mais ont besoin de se concentrer sur des zones spécifiques pour améliorer leurs performances et augmenter leurs scores de points.

IV. Conclusion de l'Exercice

Cet exercice a permis d'analyser en profondeur les performances des athlètes participant aux compétitions Olympiques et Décastar. À travers l'utilisation de l'analyse en composantes principales (ACP), nous avons réduit la dimensionnalité des données tout en préservant l'essentiel de la variance. Cela a permis de visualiser les athlètes dans un espace à deux dimensions, révélant des regroupements significatifs entre les différents sports et mettant en évidence des profils distincts.

Les résultats ont montré que les athlètes olympiques se distinguent par des performances supérieures, caractérisées par des points élevés et des rangs bas, tandis que les athlètes décastar affichent une plus grande diversité de performances. Nous avons identifié quatre profils gagnants, allant des athlètes hautement performants à ceux ayant un potentiel d'amélioration.

Ces insights fournissent une base solide pour orienter les stratégies d'entraînement et de développement des athlètes, afin d'optimiser leurs performances futures dans leurs disciplines respectives. Cette analyse peut également servir à affiner les approches d'entraînement et de préparation en fonction des besoins spécifiques de chaque groupe d'athlètes.