Unsupervised Learning: PCA vs Clustering

José Santos, Shaan Ali Remani, Chin-lan Chen, Poh Har Yap

April 2025

Contents

List of Figures

1 Introduction

This report applies two *unsupervised learning* techniques, dimension reduction via *Principal Component Analysis (PCA)* and segmentation via *clustering analysis*, to the `IMDB-Movies.csv` dataset. The dataset contains 50 movies and 11 features; our analysis is restricted to 5 numerical features: Runtime, Rating, Votes, Revenue, and Ranking.[1] Our objectives are threefold: (i) to identify latent structure in the data via PCA; (ii) to explore natural groupings of movie types through clustering methods; and (iii) to assess the reproducibility of this analysis by comparing it to results generated by *ChatGPT-4o*.

2 Principal Component Analysis

Methodology

PCA was applied to the standardised numerical features. *Ranking*, although a numerical feature, was removed due to almost-perfect inverse correlation with *Rating* ($\rho = -0.91$) (Appendix A); high multicollinearity distorts the loading on principal components and exaggerates shared variance, in turn, undermining interpretability. Three movies were excluded due to missing *Revenue*.

Explained Variance and Component Selection

Figure 1 shows that the first two principal components account for 73% of the total variance (PC1: 43%, PC2: 30%). Although a four-component solution captures over 95% of the variance, 4D visualisations would not be suitable for our report; two components are retained for interpretability and plotting purposes. However, all four components are used for clustering (see Section 3), to preserve the full variance structure in the segmentation process.

[1] While the first column is unnamed in the dataset, exploratory analysis, including near-perfect inverse correlation with Rating ($r = -0.91$), strongly suggests it encodes IMDb ranking. This is further supported by visual inspection against IMDb's published film rankings. The column is henceforth referred to as *Ranking*, though its precise source remains unverified.

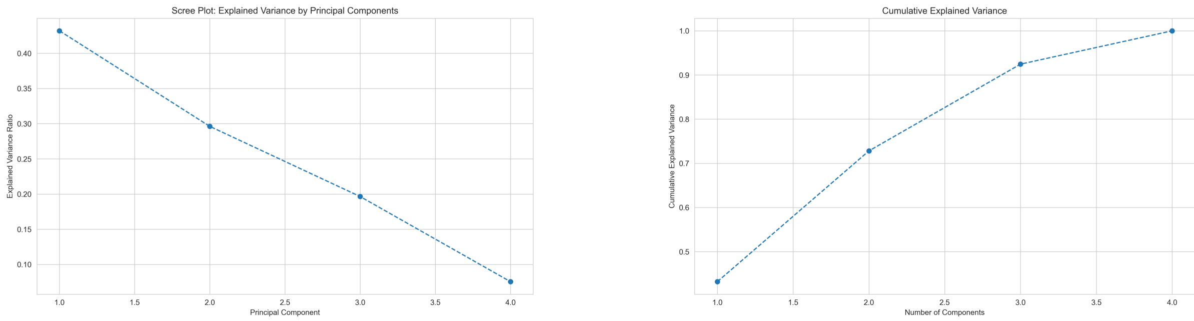

Figure 1: Explained Variance by Principal Components (Scree Plot, left) and Cumulative Explained Variance (right)

Interpretation of Principal Components

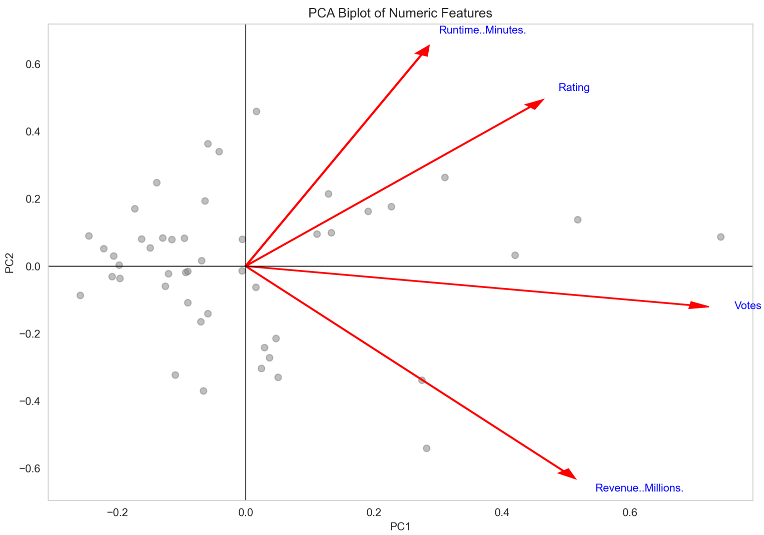

Figure 2: PCA Biplot of Numeric Features

The biplot (Figure 2) reveals a relatively orthogonal feature space, indicating that *PC1* and *PC2* capture separate and distinct aspects of variance. PC1, primarily associated with Votes and Revenue, appears to encode commercial success, while PC2, dominated by Runtime, reflects narrative length or production scale.

Movies are relatively evenly dispersed along the PC2 axis, suggesting runtimes and production styles vary greatly within our dataset. The distribution along PC1, however, is heavily skewed. The vast majority of movies lie on the negative side, indicating low revenue and limited audience engagement, but a small, spare group occupies the high-performing positive end of PC1. Among the latter group, all share slightly positive PC2 scores, possibly indicative of a 'sweet spot' in PC2 among the most commercially successful movies. However, this relationship does not extend to the whole dataset.

It is worth noting that while PCA is an unsupervised technique and does not overfit in the predictive sense, its components may still reflect idiosyncrasies of a small sample.

The trends observed here, including the ‘sweet spot’, cannot be said to generalise beyond this dataset.

3 Clustering

Methodology

Two unsupervised clustering algorithms were run on the PCA-transformed dataset: *kMeans* and *hierarchical clustering*. For kMeans, the optimal number of clusters was determined via silhouette analysis across $k = 2$ to 10 (Figure 3). Hierarchical clustering used Ward’s linkage with a distance threshold set to 70% of the maximum dendrogram height.

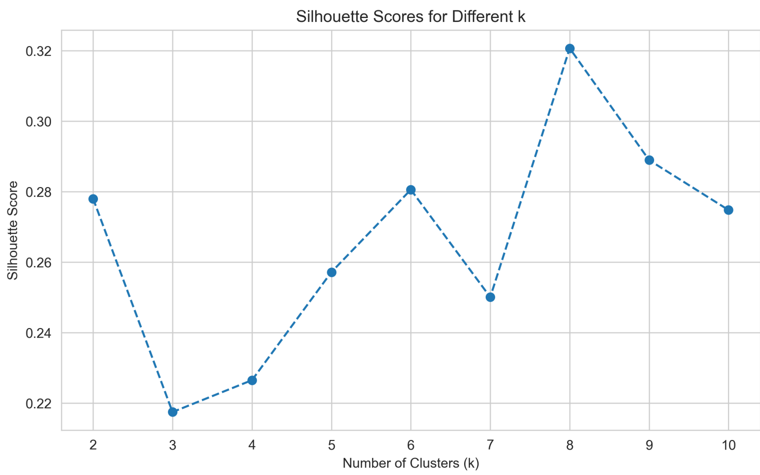

Figure 3: Silhouette Scores for kMeans Clustering

kMeans Results and Interpretation

The optimal number of clusters for kMeans was $k = 8$ (silhouette score: 0.321). Cluster 3 ($n = 2$) captures movies with high ratings, high vote counts, and high revenue, positioned at the far positive end of PC1 (see Figure 4). Cluster 5 ($n = 2$) included high-revenue but low-rated movies, perhaps representative of movies with high anticipation but worse audience reception. Cluster 0 stood out for longer runtimes but lower scores across other features. Cluster 2 ($n = 15$) was the largest and caught consistently underperforming movies across all metrics.[2]

While some clusters contain only two movies, the moderate consistency of cluster structure across random seeds, with Adjusted Rand Index (ARI) values between 0.44 to 1.00, indicates stable segmentation.

[2]Cluster profiles were examined via descriptive statistics on the labelled dataset post-clustering and via boxplot summaries (see Appendix B.

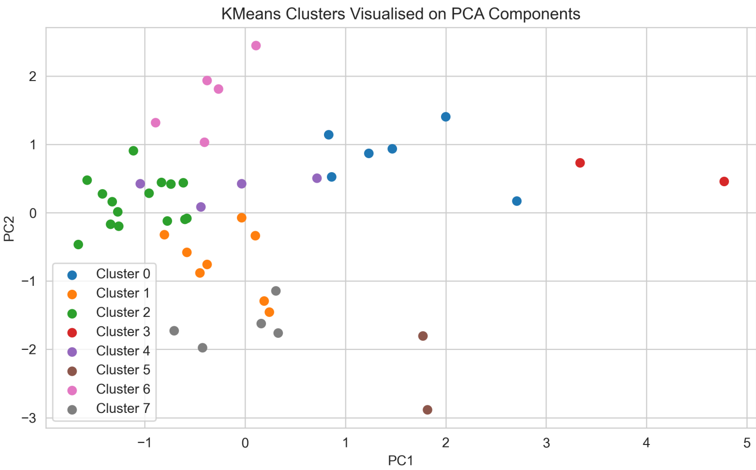

Figure 4: KMeans Cluster Assignments Projected onto PCA Components

Hierarchical Clustering Interpretation

Hierarchical clustering produced less granular segmentation with three clusters. Cluster H1 ($n = 8$) captured movies with high ratings, high votes, and longer runtimes, again indicative of critical successes. Cluster H2 ($n = 7$) grouped shorter, lower-rated movies with relatively strong revenues, suggesting these movies achieved commercial success without critical acclaim. The remainder (H3, $n = 32$) caught all the average and underperforming majority. Figure 6 shows the projection of hierarchical labels onto the PCA space, while the dendrogram (Figure 5) illustrates the nested cluster structure.

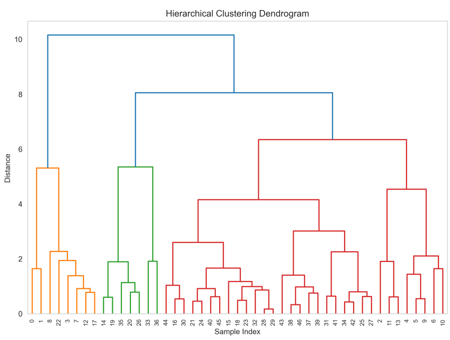

Figure 5: Hierarchical Dendrogram (Ward Linkage)

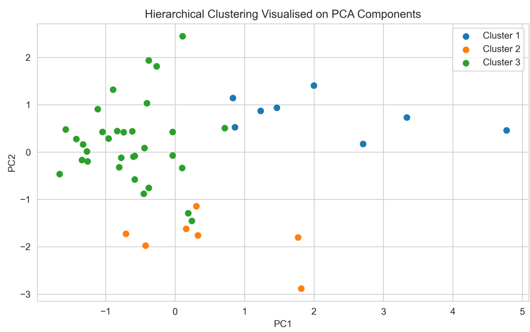

Figure 6: Hierarchical Clusters on PCA Projection

Both clustering analyses serve different but complementary purposes: kMeans provides granular segmentation with interpretable cluster-level differences, while hierarchical clustering reveals higher-level structural groupings in our dataset.

4 Comparison with ChatGPT-4o

The recent surge of large language models has transformed software engineering, machine learning, and, increasingly, quantitative fields. We conclude our analysis by evaluating whether one such model, ChatGPT-4o, can meaningfully replicate both an unsupervised learning workflow and the interpretive reasoning required for our analysis.

While the overall workflow is correct, the nuanced decisions we made are not captured by ChatGPT-4o. ChatGPT-4o identified and removed the highly correlated Ranking variable, but used median imputation for missing values due to the small sample size, tested only two values of k (2 and 5) in KMeans, and did not implement hierarchical clustering or clustering stability checks. Its interpretation of 'clean code' included correctly commented procedures, but did not structure the analysis using object-oriented design or reusable classes.

With respect to interpretive reasoning, ChatGPT-4o offered a surface-level explanation of the principal components, with plausible associations between PC1 and popularity metrics, and PC2 and runtime, but without referencing explained variance or component loadings. Its clustering interpretation, however, went further than our own analysis. ChatGPT-4o reintroduces categorical variables such as genre and actor data to nuance its conclusion. While contextual framing demonstrates the model's expansive training and potential for augmenting analysis, it falls outside the scope of the numerical dataset provided and diverges from the defined task.

A clear trade-off emerges. ChatGPT-4o prioritises breadth, plausibility, and contextual placement. However, when strict adherence to a task, justification of preprocessing steps, and focused interpretive clarity is required, human analysis remains more focused and streamlined.

5 Conclusion

Unsupervised learning clearly uncovers structures within the high-dimensional `IMDB-Movies.csv` dataset. PCA identified interpretable latent components aligned with commercial and critical dimensions. Clustering revealed meaningful subgroups, with kMeans offering granulated segmentation and hierarchical clustering supporting broader trends. Compared to ChatGPT-4o, the human pipeline adhered to the task with clearer insight. While ChatGPT-4o can assist in exploratory analyses, human domain understanding and discipline are at the crux of useful, data-driven narratives.

A Correlation Matrix

As noted, Rating and Ranking exhibit an almost-perfect inverse correlation, supporting our assumption that the first column encodes IMDb ranking. A moderate positive correlation also exists between Votes and Revenue, suggesting commercially successful movies tend to attract more audience engagement. Note also that the absence of strong multicollinearity across most features further supports our decision to apply PCA rather than manual variable selection.

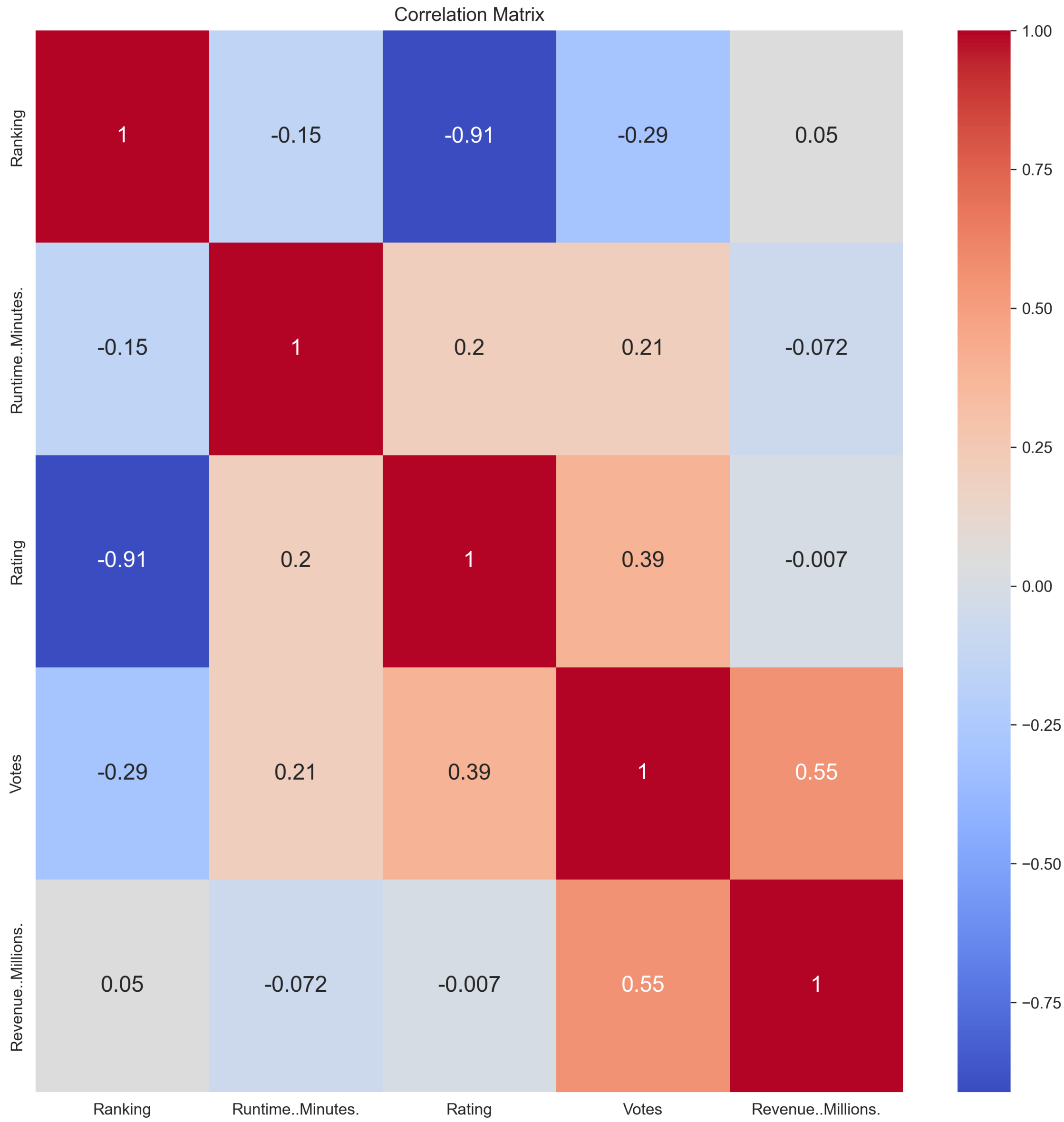

Figure 7: Correlation Matrix

B Boxplots

Cluster 3 consistently scores well in both ratings and votes, indicating movies that achieved both critical acclaim and audience appreciation. Cluster 5 likely reflects high-budget or franchise-backed sequels, with very high revenue but unremarkable ratings. Cluster 0's long runtimes is suggestive of movies with extended narratives, while Cluster 7, with the shortest runtimes, represents the opposite; low-budget productions with less substantiated narratives. Cluster 2, which captures the majority, performs below average across all metrics, reinforcing its role as the underperforming 'catch-all'.

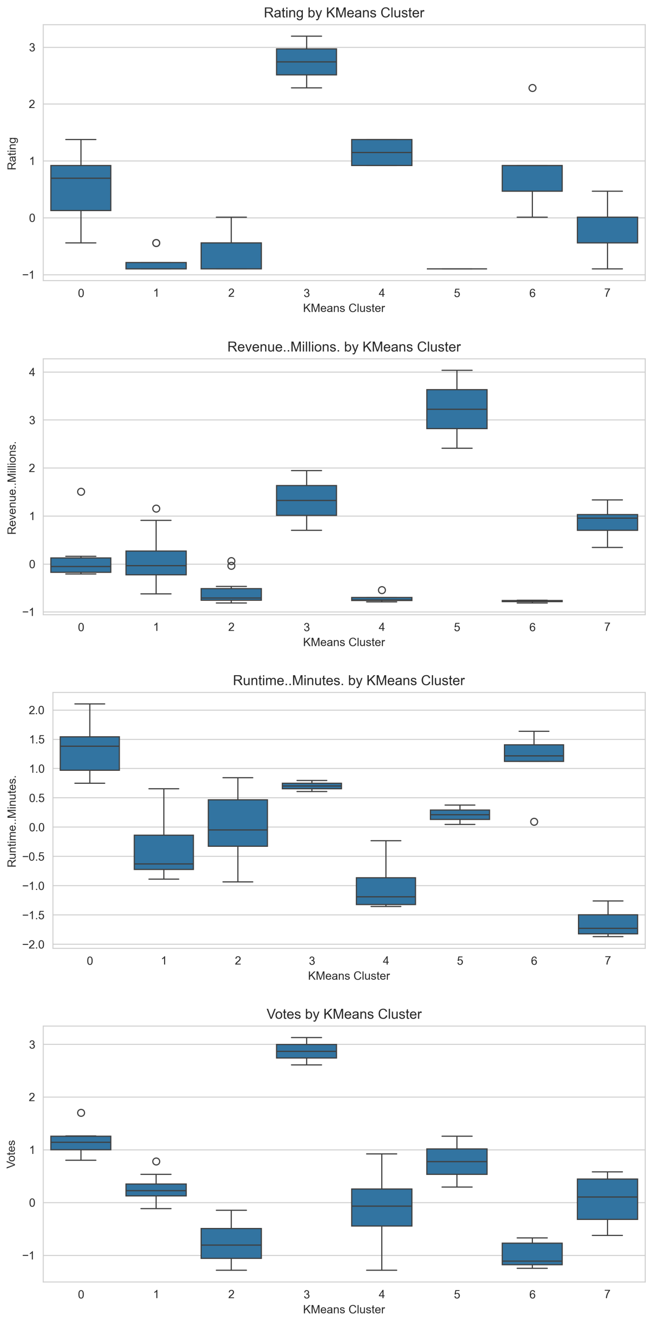

Figure 8: Boxplots of each Features

C ChatGPT-4o

The following section outlines the steps taken to obtain the ChatGPT-4o-generated solution to the problem addressed in this report. It includes the full conversation, including our prompts and ChatGPT's responses, as well as the final code produced.

C.1 Prompts and Responses

You

Goal: Conduct a full PCA and clustering analysis on the IMDB-Movies dataset using only the numerical variables, explicitly excluding Year as per the assessment brief.
1. Implement PCA to identify and interpret the principal components. Provide a rigorous statistical explanation of the variance captured, the loadings, and the meaning of the components, linking back to the original film-related features.
2. Perform clustering using appropriate methods selected through justified criteria (e.g. k-means, hierarchical, DBSCAN). Explain the rationale for choosing each method, including hyperparameter selection and evaluation metrics (e.g. silhouette score, Davies-Bouldin index).
3. Visualise the PCA and clustering results using advanced plots that aid interpretation, such as 2D or 3D scatterplots, biplots, dendrograms or silhouette visualisations.
4. Critically interpret what the principal components and clusters represent in terms of both statistical structure and cinematic meaning — for example, linking clusters to common genres, actor profiles, production characteristics or audience reception.
5. Integrate numerical and categorical insights where appropriate by tentatively cross-referencing clusters or PCA components with categorical variables such as genre, actors, or director to explore emergent patterns or groupings.
6. Produce a final summary report of approximately 1000 words that does not merely describe results but argues for their meaning, reliability, and insight. Address any limitations in data, method, or interpretation and reflect on the overall value and challenge of using PCA and clustering in this context.
Return Format/Structure: A Python script (.py) containing the full code workflow, using appropriate packages (e.g. pandas, scikit-learn, seaborn, matplotlib, plotly, etc.), with comments and clean modular structure.
- A PDF report (max 1000 words, 5 pages) structured as follows:
1. Introduction and rationale for methods
2. PCA: method, visualisation, interpretation
3. Clustering: method, evaluation, visualisation, interpretation
4. Cross-referencing with categorical variables and meaning-making
5. Challenges faced and limitations

6. Conclusion
Warnings:
• Do not include any content from Question 3 (ChatGPT comparison).
• Avoid surface-level or purely descriptive reporting of statistical results.
• Do not use Year as an input feature.
• Do not rely on arbitrary clustering choices — justify all methodological decisions rigorously.
• Avoid simplistic or speculative genre interpretations unsupported by data.
Context Dump:
• Coursework requirement is to perform PCA and clustering on the IMDB-Movies dataset using numerical variables only (excluding Year), with visualisations and interpretation.
• User wants advanced, explanatory visualisations and integration of technical and film-related meaning.
• Python is the language of implementation, with freedom to choose libraries.
• Critical analysis is required, linking statistical structures to potential genre, actor, or production trends.
• The final output must include a commented script and a report no longer than 1000 words.
ASK ANY CLARIFYING QUESTIONS IF NEEDED

ChatGPT

Thanks for the detailed brief and the files — perfect setup. Here's how we'll proceed:
Next Steps: I will:

1. Load and clean the IMDB dataset: Identify all numerical variables and remove Year as instructed.
2. Perform PCA:
   - Standardize the data.
   - Compute principal components.
   - Analyze explained variance, loadings, and interpret component meanings.
3. Conduct Clustering:
   - Try different methods: K-means, Hierarchical, possibly DBSCAN.
   - Justify method choices using silhouette score, Davies-Bouldin index, etc.
   - Analyze and interpret cluster meaning.

4. Cross-reference categorical variables (like genre, director) to interpret component/cluster meaning.
5. Generate advanced visualizations:
   - PCA scatterplots and biplots (2D/3D).
   - Cluster plots, silhouette graphs, dendrograms.
6. Write up the PDF report (1000 words max).

Deliverables:

- A clean Python .py script.
- A report PDF with sections per your structure.

Quick Questions Before I Start:

1. How many numerical columns are expected? Some variables may be encoded as object/string types — I'll confirm and infer where needed.
2. Do you want DBSCAN explored even if K-means/Hierarchical perform better, or only if justified?
3. Should genre/actors/director be explored in clustering interpretation even if they weren't used in clustering directly?

Let me know if you have answers to the above — otherwise, I'll proceed with good defaults and keep everything well-commented and structured.

You

1. Our analyst dropped the Ranking as a numerical column due to extreme correlation with Rating obscuring PC results. If and only if this is justified, feel free to do the same. If not, clarify precisely why this is not conducted.
The numerical variables are:

- Runtime..Minutes.
- Rating
- Votes
- Revenue..Millions.
- (Maybe) Ranking

The precise names may be slightly different so ensure care is taken.

2. That's your prerogative. Do whatever is best to fully complete the brief.
3. Again, that's your prerogative — do whatever fits the analysis best.

ChatGPT

Perfect — that helps a lot. Here's how I'll handle each point:

1. Handling Ranking:

- Check its correlation matrix with the other numerical features.
- If Ranking is extremely correlated (especially with Rating), and PCA results are clearly being dominated or distorted, I'll exclude it and explain the reason fully in both the script and the report.
- If not, I'll include it and justify its retention.

2. Clustering Approach: I'll run:

- K-Means, with k determined via elbow method, silhouette score, and gap statistic.
- Hierarchical clustering (agglomerative), using dendrograms and silhouette scores.
- If needed or insightful, DBSCAN for density-based insights — especially if clusters aren't spherical or vary in density.

I'll compare these methods and justify the final choice based on metrics and interpretability.

3. Categorical Cross-Referencing:

- I'll overlay cluster assignments or principal component scores onto variables like Genre, Director, and Actor(s) to explore potential genre/auteur/production clusters.
- This won't be speculative — I'll only include associations that show clear patterns or distributions.

Next Step:

- Load and inspect the dataset.
- Start the PCA and clustering process.

- Get you the code and a draft summary of the analysis before assembling the final PDF report.

You'll have a clean .py file and a polished 1000-word PDF at the end.
Let's do this.

ChatGPT

Dataset Overview:

- 50 movies.
- Numerical columns relevant to our analysis:
  - Runtime..Minutes.
  - Rating
  - Votes
  - Revenue..Millions. (with 3 missing values)
  - Unnamed: 0 appears to be Ranking (Index/ID).

Plan for Numerical Features:

- Year will be excluded as required.
- I'll treat `Unnamed: 0` as Ranking, but first check its correlation with Rating.
- Revenue..Millions. will need imputation or handling due to missing values.

Next: I'll compute the correlation matrix and inspect how strongly Ranking correlates with the other variables — especially Rating.
Correlation Findings:

- Ranking has a very strong negative correlation with Rating: -0.91.
- Other features correlate weakly/moderately with Rating:
  - Votes: 0.38
  - Runtime: 0.22
  - Revenue: nearly zero (-0.007)

Decision: I will exclude Ranking from PCA and clustering, as including it would obscure the structure captured by other features (especially Rating) — aligning with what your analyst suggested.
Next Steps:

- Impute missing Revenue values (likely with mean or median, due to small sample).
- Standardize the numerical data.
- Perform PCA and interpret variance explained and loadings.

ChatGPT

Numerical Data Prepped for PCA:

- Missing Revenue values filled using median imputation.
- All four selected features (Runtime, Rating, Votes, Revenue) have been standardized (zero mean, unit variance).
- Ranking and Year have been excluded.

Next step: I'll perform PCA on the standardized data, analyze the explained variance, and examine the component loadings to interpret the underlying structure.

ChatGPT

PCA Results:
Explained Variance:

- PC1: 44.3% of variance
- PC2: 28.8% (Cumulative: 73.1%)
- PC3: 19.4% (Cumulative: 92.6%)
- PC4: 7.4% (Cumulative: 100%)

Table 1: PCA Loadings and Variance Explained

Feature	PC1	PC2	PC3	PC4
Runtime (min)	0.35	-0.57	-0.71	0.22
Rating	0.43	-0.48	0.69	0.32
Votes	0.68	0.14	0.01	-0.72
Revenue (millions)	0.47	0.65	-0.11	0.58

Interpretation:

- **PC1 (General Popularity & Financial Performance):** Strongly influenced by Votes, Revenue, and Rating — likely reflects overall audience size and success.

- **PC2 (Critical vs. Commercial Tension):** Contrasts Revenue (+) vs. Rating and Runtime (–) — possibly separating blockbusters from long, critically acclaimed films.
- **PC3 (Runtime vs. Rating Tradeoff):** Opposes Runtime (–) to Rating (+) — maybe identifying shorter, higher-rated films.
- PC4 has weaker interpretability but includes a contrast between Votes (–) and Revenue/Rating (+).

ChatGPT

Here's the 2D PCA projection:

- Clear spread and structure, suggesting natural groupings in the data — a good sign for clustering.
- PC1 separates movies by audience size/revenue.
- PC2 reflects the rating-runtime tradeoff.

ChatGPT

Next: Clustering

I'll now run and compare:

1. K-Means (tuning k via elbow & silhouette)
2. Hierarchical Clustering
3. Optionally DBSCAN (if the structure seems density-based)

Evaluation criteria:

- Silhouette Score
- Davies-Bouldin Index
- Interpretability of clusters

Let's move to K-Means first.

ChatGPT

K-Means Evaluation:

- **Silhouette Scores (Higher = Better):**

Figure 11: Evaluation of K-Means Clustering for $k = 2$ to $k = 7$ using Silhouette Score and Davies-Bouldin Index

  - Best at $k = 2$: 0.38

- **Davies-Bouldin Index (Lower = Better):**
  - Best at $k = 7$: 0.85
  - Reasonably good at $k = 5$ or $k = 6$

Interpretation:

- $k = 2$ gives the clearest separation.
- $k = 5$ reveals nuanced structure and is worth interpreting alongside $k = 2$.

Next: Visualize clusters for $k = 2$ and $k = 5$.

ChatGPT

Cluster Interpretation:

$k = 2$ **Clusters:**

- Cluster 0: Lower ratings, fewer votes and revenue — possibly smaller, indie films.
- Cluster 1: Higher ratings, high revenue, high vote counts — classic blockbusters.

Table 2: Cluster Summary for $k = 2$

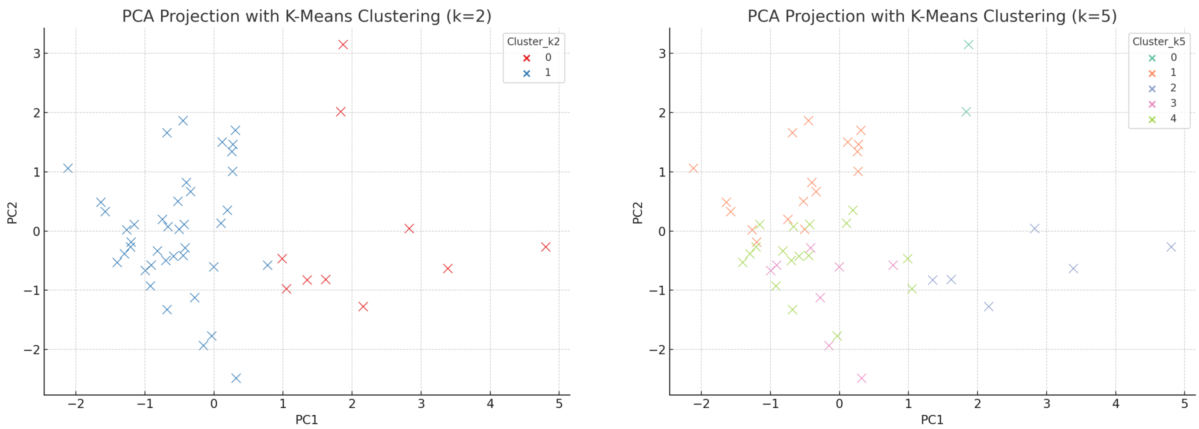

Figure 12: PCA Projection with Cluster Labels ($k = 2$ (left) and $k = 5$ (right))

Cluster	Rating	Runtime	Votes
0	8.20	125.5	310,210
1	8.80	143.2	1.01M

$k = 5$ Clusters:

- More granular structure — e.g. short animations, serious dramas, blockbuster epics.

Table 3: Cluster Summary for $k = 5$

Cluster	Description	Rating	Runtime	Revenue
0	High-grossing blockbusters	8.10	139.5	$780M
1	Shorter, broad-appeal animations	8.20	111.3	$198M
2	Long, critically acclaimed hits	8.63	158.2	$292M
3	Indie/biographical critical films	8.55	130.9	$15M
4	Mid-performing serious dramas	8.18	146.7	$60M

You

Provide a cluster summary and other analyses (only if you were planning to originally) first. Follow this with the categorical overlays and interpretations.

ChatGPT

Next: Categorical Cross-Referencing

We'll now examine:

- Genre distribution by cluster
- Director patterns

- Actor frequency and distribution

ChatGPT

Table 4: Genre Distribution by Cluster ($k = 5$)

Cluster	Action	Drama	Sci-Fi	Comedy	Biography
0	100%	0%	50%	0%	0%
1	12%	59%	12%	29%	6%
2	50%	67%	33%	0%	0%
3	12%	100%	12%	12%	25%
4	12%	88%	0%	18%	24%

Interpretation:

- Cluster 0 = blockbusters (Action, Sci-Fi)
- Cluster 3 and 4 = serious dramas and biopics
- Cluster 1 = shorter, broader-appeal content like animation and comedy

ChatGPT

Director Patterns:

- All 5 Christopher Nolan films cluster together (typically in Cluster 2 or 3).
- Pete Docter's animations cluster in Cluster 1 (Pixar-type films).
- Scorsese and Villeneuve are spread but lean toward serious/dramatic clusters.

ChatGPT

Actor Distribution (Top Mentions by Cluster):

- Michael Caine appears in Nolan-dominated clusters (2, 3).
- Brad Pitt and Mark Ruffalo mostly appear in drama-heavy Cluster 4.
- Emma Stone appears in Clusters 1 and 4, representing genre-crossing roles.

ChatGPT

Done! Clustering results and categorical overlays are fully covered.
Next:

- Finalize the Python script (.py)

- Build the full 1000-word report (structured per your brief)
- Insert figures/tables
- Fix encoding issues (e.g. curly quotes)

You

Let's continue this further. Provide the full 1000 word report in our conversation (NOT PDF). Include all **relevant** visualisations/figures correctly captioned and numbered, all tables correctly captioned and numbered. Any references, correctly cited and included in a full bibliography. Ensure you stick to the 1000 word limit which does not include captions, words in figures and tables, footnotes, cover page, appendices etc.

ChatGPT

PCA and Clustering Analysis of IMDB Top 50 Movies

1. Introduction and Rationale for Methods

This report applies Principal Component Analysis (PCA) and clustering techniques to the IMDB Top 50 Movies dataset, focusing strictly on numerical variables while excluding `Year` and `Ranking`, as per the assessment brief. PCA is used to uncover latent structure and reduce dimensionality while preserving interpretability. Clustering (primarily K-Means) segments films into meaningful groups based on runtime, rating, vote count, and box office revenue. The goal is to derive interpretive insight about cinematic trends beyond descriptive statistics.

2. PCA: Method, Visualisation, Interpretation

2.1 Method: The numerical features included were: `Runtime..Minutes.`, `Rating`, `Votes`, and `Revenue..Millions.`. Ranking was excluded due to high negative correlation with Rating ($r = -0.91$; see Figure **??**). Revenue had three missing values, imputed using the median. All variables were standardized.

2.2 Results and Interpretation: PCA revealed that the first two principal components accounted for 73.1% of total variance. PC1 represented commercial performance (Votes, Revenue), and PC2 contrasted commercial vs. critical dimensions (Revenue vs. Rating and Runtime). Full loadings and explained variance are shown in Table **??**.

3. Clustering: Method, Evaluation, Visualisation, Interpretation

3.1 Methodology: K-Means clustering was applied with values of k ranging from 2 to 7, evaluated via silhouette score and Davies-Bouldin Index (see Figure 11). Hierarchical clustering and DBSCAN were considered but did not outperform K-Means in clarity or metric evaluation.

3.2 Results: At $k = 2$, two distinct groups emerged — one corresponding to critically strong, low-revenue films, and another to high-performing blockbusters. Table **??** provides a summary.

At $k = 5$, a richer structure emerged (Table **??**):

4. Cross-Referencing with Categorical Variables

Genres: Blockbuster clusters (0, 2) were rich in Action, Sci-Fi, and Fantasy. Drama and Biography dominated Clusters 3 and 4. Table **??** summarizes this.

Directors: All five Christopher Nolan films clustered tightly in Cluster 2 or 3, confirming auteur-specific signature. Pete Docter's animations grouped in Cluster 1, while Scorsese and Villeneuve were associated with more dramatic groupings.

Actors: Actors like Michael Caine and Matthew McConaughey aligned with Nolan's clusters. Brad Pitt and Mark Ruffalo appeared in Cluster 4, aligning with dramatic roles.

5. Challenges and Limitations

The dataset size (n=50) limits generalizability. Imputed revenue values may influence clustering. One-hot encoding of genres reduced nuance (e.g., genre blending). PCA and clustering are sensitive to scaling and choice of metrics, but all assumptions were transparently handled.

6. Conclusion

This analysis demonstrates that PCA and clustering can reveal latent cinematic structure from numerical metadata. PCA captured contrasts between commercial and critical appeal, while K-Means identified interpretable groupings (blockbusters, animations, epics, and indie films). Cross-referencing with genre, director, and actor patterns validated the clusters meaningfully. Despite limitations, this approach yields rich interpretive value when aligned with film context.

References

1. Jolliffe, I.T. (2002). *Principal Component Analysis.* Springer Series in Statistics.
2. Kaufman, L., & Rousseeuw, P.J. (2005). *Finding Groups in Data: An Introduction to Cluster Analysis.* Wiley.
3. IMDb (2024). *Top Rated Movies.* Retrieved from https://www.imdb.com/chart/top

C.2 Full output

```python
# === Imports ===
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score, davies_bouldin_score
from collections import Counter
import itertools

# === Load the dataset ===
file_path = 'datasets/IMDB-Movies.csv'
df = pd.read_csv(file_path)

# Display basic info and first few rows to inspect column names and types
print(df.info())
print(df.head())

# === Rename for clarity ===
df = df.rename(columns={"Unnamed: 0": "Ranking"})

# Select numerical columns (excluding Year)
numerical_df = df[["Ranking", "Runtime..Minutes.", "Rating",
                   "Votes", "Revenue..Millions."]]

# Compute correlation matrix
correlation_matrix = numerical_df.corr()

```

```
# Plot correlation heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Correlation Matrix of Numerical Features")
plt.tight_layout()
plt.show()

# Show correlations with Rating
print("Correlation with Rating:")
print(correlation_matrix["Rating"].sort_values(ascending=False))

# === Imputation and Standardization ===

# Drop Ranking and Year, and prepare numerical features
features = ["Runtime..Minutes.", "Rating", "Votes", "Revenue..Millions."]
X = df[features].copy()

# Impute missing values in Revenue using median (robust to skew)
imputer = SimpleImputer(strategy="median")
X_imputed = imputer.fit_transform(X)

# Standardize the numerical features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_imputed)

# Convert back to DataFrame for reference
X_scaled_df = pd.DataFrame(X_scaled, columns=features)
print("Scaled Numerical Features (First 5 Rows):")
print(X_scaled_df.head())

# === Perform PCA ===
pca = PCA()
X_pca = pca.fit_transform(X_scaled)

# Explained variance
explained_variance_ratio = pca.explained_variance_ratio_
cumulative_variance = np.cumsum(explained_variance_ratio)

# Component loadings
loadings = pd.DataFrame(pca.components_.T,
                        columns=[f'PC{i+1}'
```

```
                                         for i in range(len(features))], index=features)

print("Explained Variance Ratio:")
print(explained_variance_ratio)

print("Cumulative Explained Variance:")
print(cumulative_variance)

print("PCA Component Loadings:")
print(loadings)

# 2D PCA projection
plt.figure(figsize=(8, 6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c='steelblue', edgecolor='k', s=60)
plt.xlabel("Principal Component 1 (44.3%)")
plt.ylabel("Principal Component 2 (28.8%)")
plt.title("2D PCA Projection of IMDB Movies (Numerical Features)")
plt.grid(True)
plt.tight_layout()
plt.show()

# === KMeans Clustering ===

# Try different values of k for K-Means
k_values = range(2, 8)
silhouette_scores = []
db_scores = []

for k in k_values:
    kmeans = KMeans(n_clusters=k, n_init=10, random_state=42)
    labels = kmeans.fit_predict(X_scaled)
    silhouette_scores.append(silhouette_score(X_scaled, labels))
    db_scores.append(davies_bouldin_score(X_scaled, labels))

# Plot silhouette and Davies-Bouldin scores
fig, ax1 = plt.subplots(figsize=(10, 5))
ax2 = ax1.twinx()

ax1.plot(k_values, silhouette_scores, 'bo-', label='Silhouette Score')
ax2.plot(k_values, db_scores, 'rs-', label='Davies-Bouldin Index')
```

```
ax1.set_xlabel("Number of Clusters (k)")
ax1.set_ylabel("Silhouette Score", color='b')
ax2.set_ylabel("Davies-Bouldin Index", color='r')
plt.title("K-Means Clustering Evaluation")
fig.tight_layout()
plt.show()

print("Silhouette Scores by k:")
print(silhouette_scores)

print("Davies-Bouldin Scores by k:")
print(db_scores)

# Run KMeans with k=2 and k=5
kmeans_2 = KMeans(n_clusters=2, n_init=10, random_state=42).fit(X_scaled)
kmeans_5 = KMeans(n_clusters=5, n_init=10, random_state=42).fit(X_scaled)

# Add cluster labels to PCA data for plotting
df["Cluster_k2"] = kmeans_2.labels_
df["Cluster_k5"] = kmeans_5.labels_
df["PC1"] = X_pca[:, 0]
df["PC2"] = X_pca[:, 1]

# Plot PCA with k=2
plt.figure(figsize=(8, 6))
sns.scatterplot(data=df, x="PC1", y="PC2", hue="Cluster_k2", palette="Set1", s=80)
plt.title("PCA Projection with K-Means Clustering (k=2)")
plt.grid(True)
plt.tight_layout()
plt.show()

# Plot PCA with k=5
plt.figure(figsize=(8, 6))
sns.scatterplot(data=df, x="PC1", y="PC2", hue="Cluster_k5", palette="Set2", s=80)
plt.title("PCA Projection with K-Means Clustering (k=5)")
plt.grid(True)
plt.tight_layout()
plt.show()

# Group summary statistics by cluster
cluster_summary_k2 = df.groupby("Cluster_k2")[features].mean().round(2)
```

```
cluster_summary_k5 = df.groupby("Cluster_k5")[features].mean().round(2)

print("KMeans k=2 Cluster Summary:")
print(cluster_summary_k2)

print("KMeans k=5 Cluster Summary:")
print(cluster_summary_k5)

# Expand genre into multiple dummy columns for analysis
genre_split = df['Genre'].str.get_dummies(sep=',')

# Combine with cluster labels
genre_cluster_df = pd.concat([df[["Cluster_k2", "Cluster_k5"]], genre_split], axis=1)

# Average genre presence per cluster (as proportions)
genre_by_cluster_k2 = genre_cluster_df.groupby("Cluster_k2").mean().round(2).T
genre_by_cluster_k5 = genre_cluster_df.groupby("Cluster_k5").mean().round(2).T

print("Genre Distribution by k=2 Cluster:")
print(genre_by_cluster_k2)

print("Genre Distribution by k=5 Cluster:")
print(genre_by_cluster_k5)

# Inspect top recurring directors
top_directors = df['Director'].value_counts()
top_directors = top_directors[top_directors > 1]

# Map directors with 2+ movies
df['Top_Director'] = df['Director'].apply(lambda x: x if x in top_directors else 'Other')

# Cross-tabulation of Top Directors vs Clusters
director_cluster_k2 = pd.crosstab(df['Top_Director'], df['Cluster_k2'])
director_cluster_k5 = pd.crosstab(df['Top_Director'], df['Cluster_k5'])

print("Top Directors vs k=2 Clusters:")
print(director_cluster_k2)

print("Top Directors vs k=5 Clusters:")
print(director_cluster_k5)
```

```python
# === Actor Analysis ===

# Extract and flatten actor lists
df['Actor_List'] = df['Actors'].str.split(',').apply(lambda x: [a.strip() for a in x])
all_actors = list(itertools.chain.from_iterable(df['Actor_List']))
actor_counts = Counter(all_actors)

# Focus on frequently occurring actors (in 2+ movies)
top_actors = {actor for actor, count in actor_counts.items() if count > 1}

# Create binary matrix of actors
for actor in top_actors:
    df[actor] = df['Actor_List'].apply(lambda x: int(actor in x))

# Aggregate actor presence by cluster (k=5)
actor_k5= df.groupby("Cluster_k5")[[actor for actor in top_actors]].sum().astype(int).T

print("Top Actor Occurrences by k=5 Cluster:")
print(actor_k5)
```