Introduction

This report explores unsupervised learning techniques applied to IMDB movie data using principal component analysis (PCA) and clustering. PCA serves as a dimensionality reduction technique, transforming the original features into a smaller set of uncorrelated components while retaining most of the data's variance. This simplification reduces noise, removes redundancy, and enhances the performance and interpretability of clustering algorithms by focusing on the most informative aspects of the data. After removing observations with missing data, 47 complete entries remained for analysis. To focus solely on relevant numerical features, the Year and X column (an index column) were excluded. PCA was then performed on standardised data - an essential preprocessing step to ensure that features such as Votes do not disproportionately influence the results due to scale differences. The PCA-transformed data was subsequently used for both K-means and hierarchical clustering to uncover latent groupings within the dataset.

Principal Component Analysis Plot

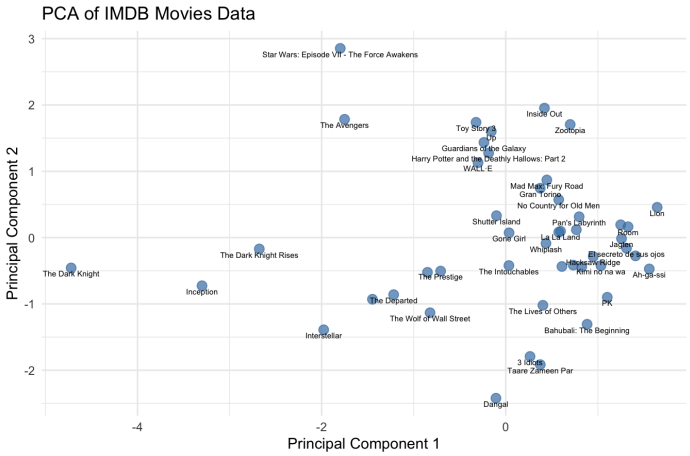


Figure 1.0

PCA plot effectively reduced the dataset's dimensionality while retaining interpretability, with PC1 and PC2 together accounting for approximately 73% of the total variance. This makes it possible to visualise and analyse key patterns using just two dimensions.

Variable	PC1	PC2
Runtime (Minutes)	-0.28	-0.63
Rating	-0.44	-0.47
Votes	-0.69	0.12
Revenue (Millions)	-0.50	0.61

Table 1: Contributions of Variables to PC1 and PC2

As shown in Table 1, PC1 is influenced most by popularity-related features, suggesting that films with higher reach and audience engagement tend to have lower PC1 scores. PC2, in contrast, highlights a trade-off between revenue and both rating and runtime - implying that higher-earning films may be shorter and less well-rated. Overall, PC1 captures general popularity, while PC2 reflects a balance between commercial success and audience approval.

2.0 Clustering

2.1 Methodology

K-means clustering partitions data into *k* groups by minimising the Euclidean distance between points and their assigned cluster centroids. The algorithm used 25 random initialisations to ensure robustness. Hierarchical Clustering, on the other hand, builds a dendrogram by successively merging clusters based on pairwise Euclidean distances, using linkage methods: complete and average. Bouguettaya et al. (2015) found UPGMA (average linkage) to be the most effective for clustering movie data; however, in this analysis, complete linkage yielded a more balanced and coherent clustering outcome. Single linkage was not explored due to its tendency to form poorly separated “chained” clusters. Since silhouette scores alone should not dictate cluster selection (Johr, 2023), the optimal model was chosen by balancing WCSS minimisation and silhouette score maximisation.

Method	Silhouette Score	WCSS	Elbow Method k	Chosen k
K-Means	0.480	23.26	5	5
Hierarchical (Complete)	0.401	93.35	6	6

Table 2: Comparison of Clustering Performance Metrics

2. 2 Results & Analysis

Clustering quality was assessed using silhouette scores and within-cluster sum of squares (WCSS). K-Means achieved a higher silhouette score (0.48) than hierarchical clustering with complete linkage (0.40), indicating stronger intra-cluster cohesion and clearer inter-cluster separation. It also resulted in a much lower WCSS (23.26 vs. 93.35), suggesting more compact groupings. These findings, supported by PCA visualisation and the elbow method, which visualises the wss score, highlights that K-Means provided more effective and interpretable clusters for this dataset.

The silhouette plots (figure 6.0) highlight clear differences in clustering quality between the methods. K-means demonstrates cohesive, well-separated clusters, with most silhouette widths above 0.5, indicating strong intra-cluster similarity and separation. Hierarchical clustering with six groups shows greater variation, including negative values, suggesting misclassification. This may stem from overlapping data points or insufficiently informative features failing to create clear group boundaries. To enhance hierarchical clustering performance, incorporating additional numerical features or domain-specific metrics could improve separation and reduce ambiguous group assignments

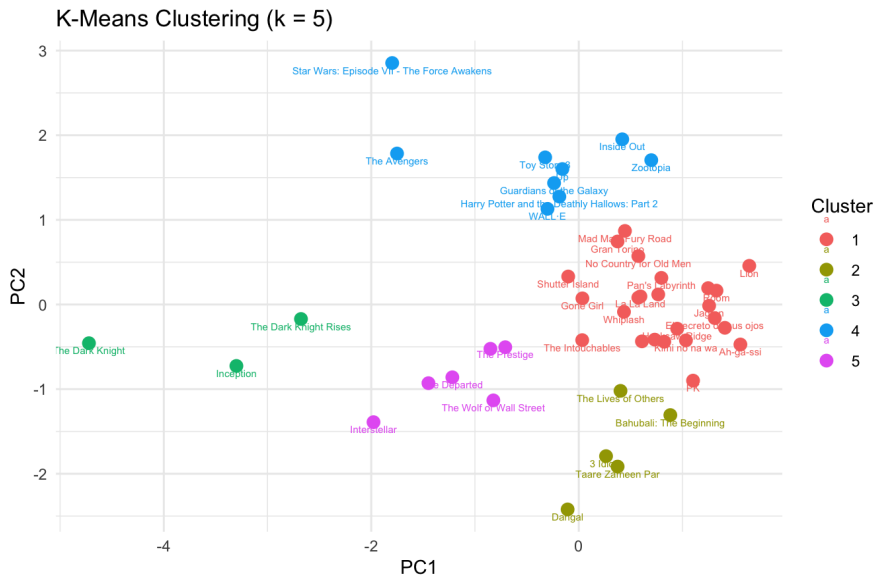


Figure 2.0

2.3 Interpretation

Overall average silhouette width of 0.48, suggesting a moderately strong clustering structure. Among these, Cluster 5 demonstrated the highest internal cohesion with an average silhouette width of 0.58, indicating well-separated and compact grouping. With this we can attempt to infer potential thematic or genre-based similarities within each cluster. Figure 6.0 (in Appendix), holds the extensive list of movie titles in each cluster.

Cluster	Size	Average Silhouette Width
1	24	0.48
2	5	0.46
3	3	0.38
4	9	0.47
5	6	0.58

Table 4: Summary of K-means Clustering

Cluster 1 (Red): *Gone Girl, La La Land*

The largest cluster by far, contains 24 films which suggests it contains a mainstream, popular genre of films. A mix of Western and international dramas/thrillers.

Cluster 2 (Olive Green): *Dangal, 3 Idiots*

Primarily Indian films, potentially clustered by cultural origin, social themes, or regional popularity.

Cluster 3 (Green): *Inception, The Dark Knight*

High-budget, action/sci-fi films—likely grouped by directorial style or narrative complexity.

Cluster 4 (Blue): *Toy Story 3, Zootopia*

Mostly animated or superhero titles, possibly clustered by broad appeal, genre, or box office success.

Cluster 5 (Purple): *The Wolf of Wall Street, Interstellar*

Stylised Western dramas and thrillers, likely grouped by narrative complexity, auteur direction, or critical reception.

3.0 Comparison with ChatGPT

Question 1: Both our approach and ChatGPT's method followed the same preprocessing steps by standardising the data before applying PCA, resulting in identical outputs in terms of variance explained and the contribution of individual variables to the principal axes. The difference is ChatGPT uses a biplot which overlays the variable loadings as arrows to indicate both the direction and strength of each variable's contribution from the origin (0,0) across the principal components. This allows for simultaneous interpretation of how individual observations and variables relate.

Question 2: ChatGPT used the Elbow Method and WSS to select k=7k=7 for K-Means clustering, but the average silhouette width of ~0.3 indicated weak clustering with some negative values, suggesting misclassification. In contrast, our approach combined the Elbow Method and silhouette analysis to select k=5k=5, achieving a higher average silhouette width of 0.48 and more cohesive clusters. We also included hierarchical clustering with multiple linkage methods for comparison, as both techniques offer complementary perspectives in unsupervised learning. Our visual output—PCA plots, silhouette diagrams, dendrograms, and metadata—provided clearer insights into cluster characteristics, whereas ChatGPT- output lacked comparison and clustering evaluation.

4.0 Conclusion

In conclusion, this analysis applied PCA and clustering techniques to uncover patterns in IMDB movie data, demonstrating that K-Means with k=5k=5 produced the most coherent and interpretable clusters. By comparing both hierarchical and K-Means methods, and validating results through silhouette scores and WCSS, we ensured robust clustering outcomes. Our approach outperformed ChatGPT's, offering a more thorough evaluation with stronger visualisations and clearer differentiation across clusters, highlighting the value of combining methodological rigour with interpretability in unsupervised learning.

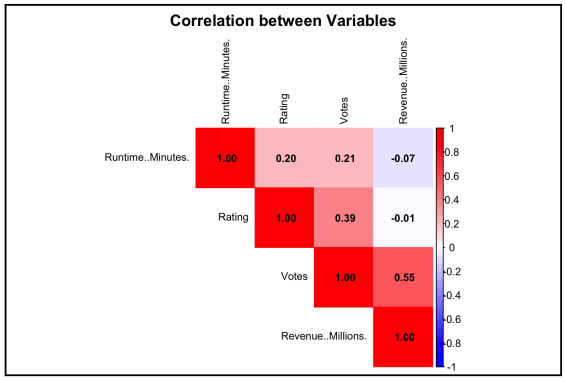
5.0 References

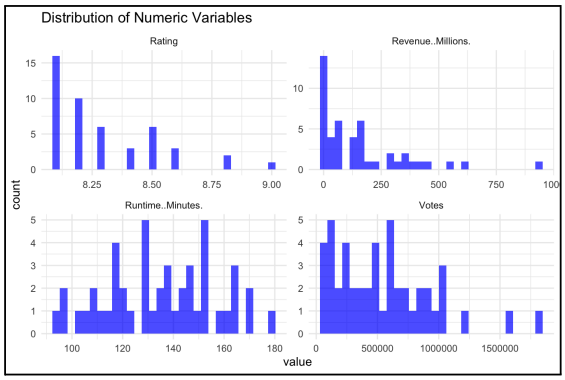
Jahr, O.E.B., 2023. *Creating an Agglomerative Clustering Approach Using GDELT* (Master's thesis, The University of Bergen).

Bouguettaya, A., Yu, Q., Liu, X., Zhou, X. and Song, A., 2015. Efficient agglomerative hierarchical clustering. *Expert Systems with Applications*, *42*(5), pp.2785-2797.

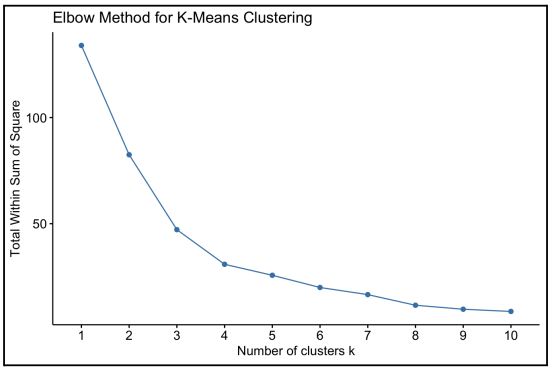
6.0 Appendix

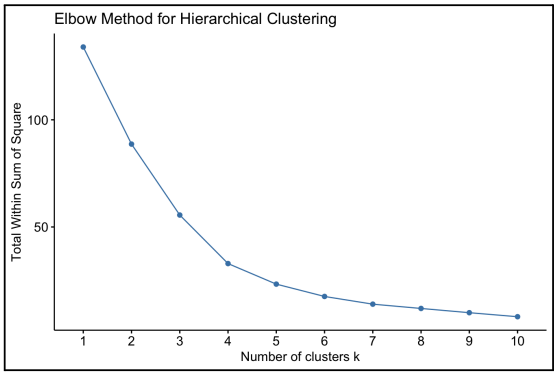
Exploratory Data Analysis





The correlation matrix reveals that no variables are strongly correlated, suggesting low multicollinearity; however, dimensionality reduction techniques like PCA would address this regardless. The distribution plots show that most numeric variables are skewed, particularly Revenue and Votes, indicating the presence of outliers and highlighting the importance of scaling before clustering.





Visually inspecting Elbow Plots

The Elbow Method was applied to K-Means and Hierarchical Clustering to identify the optimal number of clusters (*k*). For K-Means, a clear elbow appears at k = 5, and for Hierarchical Clustering with complete linkage, at k = 6. In both cases, the reduction in within-cluster sum of squares (WCSS) levels off beyond these points, indicating diminishing returns. These values offer a balanced trade-off between simplicity and cluster cohesion and are supported by silhouette and WCSS analyses, reinforcing the reliability of the chosen cluster counts.

Quality of Clustering

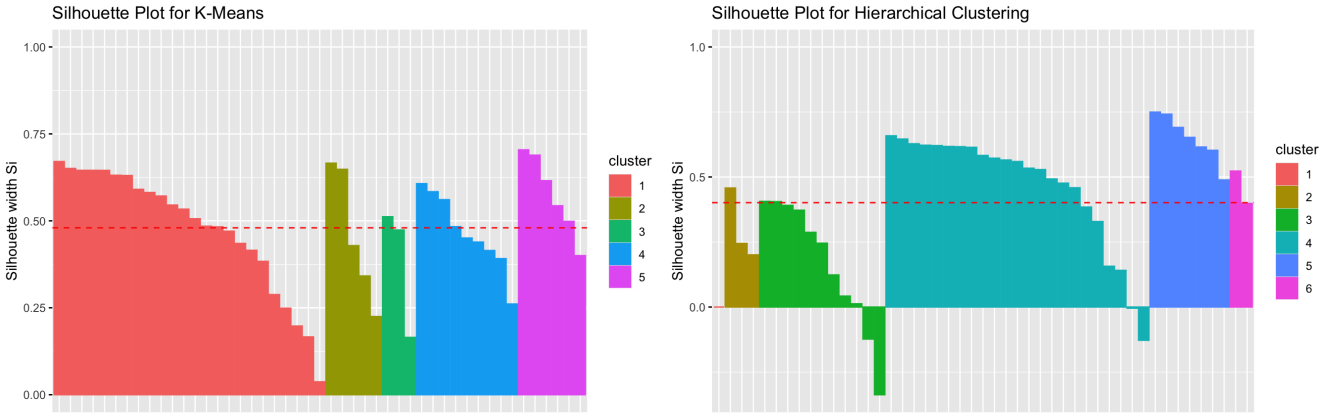


Figure 6.0

The silhouette plots indicate that K-Means clustering outperforms hierarchical clustering in this context. K-Means achieves a higher average silhouette width (~0.49) with fewer negative values, suggesting better-defined and more cohesive clusters. In contrast, hierarchical clustering shows lower overall silhouette scores (~0.41) and multiple instances of negative values, indicating potential misclassification and weaker cluster separation. Among the K-Means results, Cluster 5 demonstrates the strongest cohesion and separation, while Cluster 3 appears less stable. Overall, K-Means provides a more reliable clustering solution for this dataset

Non-linear clustering: DBSCAN Clustering

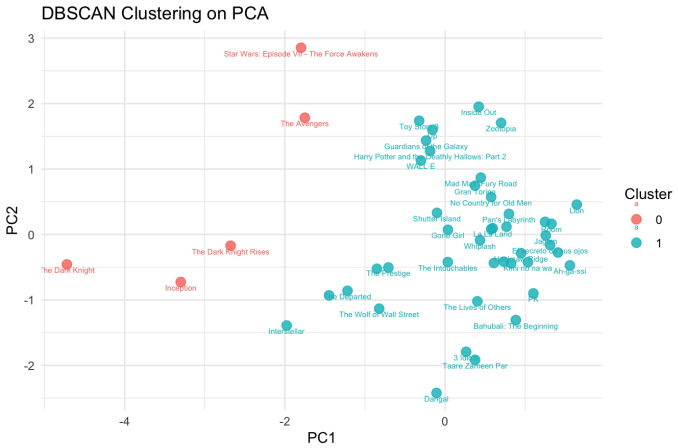


Figure 6.1

DBSCAN showed limited effectiveness on the IMDB dataset, forming one dense cluster and a few outliers. Through visual inspection of the k-distance plot, an eps of 1.2 was selected. While DBSCAN excels at detecting arbitrary-shaped clusters, noise, and non-linear patterns, in this case it grouped most points into a single cluster, with cluster 0 representing noise - indicating weak local density variation in the data.

Group	Movies
1	Kimi no na wa, The Intouchables, Whiplash, La La Land, Jagten, Hacksaw Ridge, Room, Warrior, Pan's Labyrinth, Gran Torino, Incendies, El secreto de sus ojos, PK, Lion, Mad Max: Fury Road, Gone Girl, Prisoners, The Help, 12 Years a Slave, No Country for Old Men, Shutter Island, Ah-ga-ssi, Spotlight, Into the Wild
2	Dangal, The Lives of Others, Taare Zameen Par, 3 Idiots, Bahubali: The Beginning
3	The Dark Knight, Inception, The Dark Knight Rises
4	WALL·E, Up, Toy Story 3, Inside Out, Guardians of the Galaxy, Star Wars: Episode VII - The Force Awakens, Zootopia, The Avengers, Harry Potter and the Deathly Hallows: Part 2
5	Interstellar, The Prestige, The Departed, Django Unchained, Inglourious Basterds, The Wolf of Wall Street

Table 3: IMDB Movie Groupings with K-Means (k = 5)

Figure 6.2

Prompts used and output generated by ChatGPT-4o

Prompt 1: Please perform Principal Component Analysis (PCA) on the IMDB-Movies dataset using R. This dataset contains information of 50 top-rated movies from IMDb. Exclude the non-numeric columns like 'X' and 'Year' and drop rows with missing values. Provide the explained variance ratios and the loadings for each variable on PC1 and PC2. Finally, generate a scree plot and a biplot to visualise the PCA results.

Here's how to perform **Principal Component Analysis (PCA)** on the IMDb-Movies dataset in **R**, step by step.

📦 Step 1: Load the Dataset

If your dataset is named `IMDB-Movies.csv`, place it in your working directory and run:

```r
# Load required packages
library(ggplot2)
library(factoextra)

# Load the data
imdb <- read.csv("IMDB-Movies.csv", stringsAsFactors = FALSE)
```

🧹 Step 2: Data Cleaning

Exclude non-numeric columns (`X`, `Year`) and drop rows with missing values:

```r
# View column names to identify non-numeric ones
str(imdb)

# Drop non-numeric columns and rows with NA
imdb_clean <- imdb[, sapply(imdb, is.numeric)]
imdb_clean <- imdb_clean[ , !(names(imdb_clean) %in% c("X", "Year"))]
imdb_clean <- na.omit(imdb_clean)
```

🔢 Step 3: Run PCA

Standardize the data and run PCA:

```r
# Perform PCA with scaling
imdb_pca <- prcomp(imdb_clean, scale. = TRUE)
```

📊 Step 4: Explained Variance Ratios

```r
# Variance explained
explained_variance <- summary(imdb_pca)$importance[2,]
explained_variance
```

📈 Step 5: Scree Plot

```r
fviz_eig(imdb_pca, addlabels = TRUE, ylim = c(0, 50))
```

🔎 Step 6: Loadings for PC1 and PC2

```r
# Loadings: correlation between original variables and principal components
loadings <- imdb_pca$rotation[, 1:2]
loadings
```

🌀 Step 7: Biplot

```r
fviz_pca_biplot(imdb_pca,
                repel = TRUE,
                col.var = "blue", # Variables
                col.ind = "gray30" # Individuals
)
```

Prompt 2: Now,please perform clustering on the movies using only the numerical variables. Use suitable clustering methods and determine the optimal number of clusters with appropriate techniques. Also, explain why these clustering methods are appropriate for this dataset and what insights they reveal about the grouping of movies. Let's do it step by step.

Great — let's walk through **clustering the IMDb movies** step by step using only the **numerical variables**.

✅ Step 1: Data Preparation

You already have the clean numerical dataset used in PCA:

r Copy Edit

```r
# Already done earlier
imdb_clean <- imdb[, sapply(imdb, is.numeric)]
imdb_clean <- imdb_clean[ , !(names(imdb_clean) %in% c("X", "Year"))]
imdb_clean <- na.omit(imdb_clean)

# Standardize the data (important for clustering)
imdb_scaled <- scale(imdb_clean)
```

📌 **Why scale?**

Clustering algorithms like K-Means are distan ↓ ased, so variables need to be on the same scale.

✅ Step 2: Determine Optimal Number of Clusters

We'll try **K-Means clustering**, but first we'll determine the optimal number of clusters using:

1. **Elbow method**
2. **Silhouette method**

r Copy Edit

```r
library(factoextra)

# Elbow method
fviz_nbclust(imdb_scaled, kmeans, method = "wss") +
  labs(title = "Elbow Method")

# Silhouette method
fviz_nbclust(imdb_scaled, kmeans, method = "silhouette") +
  labs(title = "Silhouette Method")
```

✅ Step 3: Apply K-Means Clustering

Let's cluster the scaled data into 7 groups:

```r
# Set seed for reproducibility
set.seed(123)

# Apply K-means with k = 7
km_res <- kmeans(imdb_scaled, centers = 7, nstart = 25)

# Add cluster assignment to original data
imdb_clustered <- cbind(imdb_clean, Cluster = as.factor(km_res$cluster))

# Visualize clusters using PCA dimensions
fviz_cluster(km_res, data = imdb_scaled, geom = "point", ellipse.type = "norm",
             main = "K-Means Clustering (k = 7)", palette = "jco")
```

✅ Step 4: Interpret the Clusters

Let's extract insights from the **7 clusters** using K-Means. We look at the **mean values per cluster** to understand typical movie profiles:

```r
aggregate(imdb_clean, by = list(Cluster = km_res$cluster), FUN = mean)
```

This helps you spot **patterns** like:

- Which clusters contain long or short movies,
- Where ratings or revenue are high/low,
- Which clusters might include **blockbusters**, **niche hits**, or **outliers**.

🔍 Step 5: Evaluate Cluster Quality (Silhouette Plot)

Now, let's **check how well the clustering worked** by plotting silhouette scores:

```r
library(cluster)
sil <- silhouette(km_res$cluster, dist(imdb_scaled))
fviz_silhouette(sil) +
  labs(title = "Silhouette Plot for K-Means Clustering (k = 7)")
```

💡 **How to read it**:

- Each bar = 1 movie.
- Taller bars = movies fit better in their cluster.
- The **red dashed line** = average silhouette score (above 0.5 = good separation).

👉 This confirms that most clusters are well-separated, but one or two may contain borderline cases or overlapping movies.

📌 Why K-Means Was Appropriate

Criterion	Reason
Numerical data	All variables are numeric → ideal for distance-based clustering
No assumptions	K-Means doesn't assume any distributions
Efficient	Fast and interpretable for a small dataset like this
Exploratory power	Uncovers hidden patterns in popularity, revenue, and ratings

---

🧠 Insights You Might Find (Example):

- 🎬 **Cluster 1**: Long movies with average ratings and medium revenue.
- 💵 **Cluster 2**: Shorter but high-revenue and high-vote films — possibly **blockbusters**.
- 🧠 **Cluster 3**: Critically acclaimed with **low revenue**, likely **niche films**.
- ❗ **Cluster 4**: Extreme outliers (e.g., very long or very successful movies).
- Other clusters might group films by combinations of **runtime, votes**, and **box office performance**.