Import the required libraries for the project

**1.Data Collection & Exploration**

* Reads the dataset into a **Pandas DataFrame** from a .csv file located on system.
* df.info() shows summary info: number of entries, column names, data types, and non-null values.
* df.head() shows the **first 5 rows** to get a preview of the data.
* Checks for **missing/null values** in each column.
* Prints only the columns where missing values exist.
* Checks how many rows are exact duplicates of others.
* Displays the count of unique/distinct entries in each column.
* Uses Seaborn to plot a bar chart showing how many Movies vs TV Shows are present.
* Finds the top 10 countries that produced the most titles.
* Plots a horizontal bar chart of these countries.
* Shows a bar chart of the top 10 most frequent ratings (like TV-MA, PG, R, etc.).
* **Clean date\_added**: Removes spaces and converts it to datetime format.
* **Extract Year**: Creates a new column year\_added from the datetime.
* **Plot**: Shows a bar chart of how many titles were added to Netflix each year.

This visualization helps you analyze **trends over time**, such as:

* When Netflix added the most content
* Whether content addition is increasing or declining over the years

**2.Data Preprocessing**

**1. Handle Missing Values**

* Fills director, cast, country with 'Unknown'.
* Converts date\_added to proper datetime format.

**2. Encode Categorical Data**

* One-hot encodes type and rating.
* Splits duration into numeric (duration\_num) and unit (duration\_type), then one-hot encodes units.

**3. Scale Numeric Data**

* Standardizes release\_year and duration\_num using StandardScaler.

**4. Extract Features with NLP**

* Applies TF-IDF to listed\_in and description (text columns) to create numeric feature vectors.

**5. Final Dataset**

* Combines all processed data into df\_final.
* Drops unnecessary columns like title, show\_id, and text fields.

**RESULT**:

* This dataframe has 116 columns, all numeric.
* It's fully cleaned and transformed — ideal for use in clustering, classification, or other machine learning models.

**3. Feature Engineering**

**1. Handle Missing Values**

* Fills missing release\_year with the median year.
* Fills missing genres (listed\_in) with 'Unknown'.

**2. Create New Features**

* Calculates how old each movie/show is based on the current year.
* Counts how many genres are listed per title (e.g., 2 for "Drama, Thriller").

**3.Convert Booleans to Integers**

* Ensures all boolean columns are stored as integers (0 or 1), for model compatibility.

**4. Encode Categorical Data**

* One-hot encodes type (Movie/TV Show) and rating (like PG, R, etc.).

**5. Scale New Features**

* Scales content\_age and genre\_count to standard format (mean=0, std=1).

**Result**

You get two engineered features (content\_age, genre\_count), and all data is clean, numeric, and scaled — ready for use in ML models.

**4. Clustering Model Selection**

**1. K Means Clustering**

* Selects only numeric features from the dataset.
* Runs K-Means clustering for K = 2 to 10.
* Plots the Elbow Curve to find the best number of clusters (where the curve bends).
* The "elbow point" shows the optimal K for clustering.

**2. HIERARCHICAL CLUSTERING**  
  
 Performs Hierarchical Clustering using the Ward method

* Creates a linkage matrix to define how clusters are merged.
* Plots a dendrogram to visualize the cluster merging process.
* Helps you decide the number of clusters by cutting the dendrogram at a certain height.

**3. DBSCAN**

* Applies DBSCAN to detect clusters based on point density.
* Adds cluster labels to the DataFrame (-1 = noise).
* Prints cluster counts.
* Plots the first two features colored by cluster label.

**5. Model Training & Optimization**

**1. silhouette score**

* Uses df\_final and selects only numeric features.
* Fills any missing values using the mean.
* Runs K-Means for K = 2 to 10 and computes the silhouette score for each.
* Plots the silhouette scores to find the best K (higher score = better clustering).
* Trains final K-Means model with K = 4 and adds cluster labels to df\_final.

**2. Linkage method**

* Performs Hierarchical Clustering using the 'ward' linkage method.
* Plots a dendrogram to visualize cluster formation.
* Fits a final model with 4 clusters and adds the labels to df\_final.

**3. PCA for 2D Visualization**

* Applies PCA to reduce features to 2D for visualization.
* Adds the 2 principal components (pca1, pca2) to df\_final.
* Plots a scatterplot of K-Means clusters in 2D space using these PCA components.

**6. Visualization & Interpretation**

**1. Heat map**

* Computes average feature values per K-Means cluster.
* Filters out less interpretable columns like PCA and TF-IDF.
* Plots a heatmap to visualize how key features vary across clusters.
* Helps you understand the characteristics of each cluster.

**2. Insights: Common Genres per Cluster**

* Adds K-Means cluster labels back to the original df to analyze genres.
* For each cluster, counts and prints the top 5 most common genres (listed\_in).
* Helps identify genre preferences within each cluster.

This output shows the **top 5 most common genres** for each K-Means cluster:

* **Cluster 0**: Focus on **international movies**, **dramas**, and **comedies**.
* **Cluster 1**: Dominated by **TV content**, especially **international**, **dramas**, and **comedies**.
* **Cluster 2**: Smaller cluster with a mix of **dramas**, **classics**, and **action**.
* **Cluster 3**: Similar to Cluster 0, but with more **documentaries**.

**7. Evaluation & Refinement**

* Runs KMeans, Hierarchical, and DBSCAN clustering.
* Evaluates each using Silhouette Score and Davies-Bouldin Index.
* Also prints KMeans inertia and number of DBSCAN noise points.
* Helps compare clustering model performance.

**RESULT**:

### ****KMeans****

* **Silhouette Score**: 0.3366 → Moderate cluster separation.
* **Davies-Bouldin Index**: 1.0070 → Lower is better; decent structure.
* **Inertia**: 33376.2376 → Total distance within clusters (lower is better).

### ****Agglomerative (Hierarchical)****

* Silhouette Score**: 0.3339 → Similar to KMeans.**
* Davies-Bouldin Index**: 0.9877 → Slightly better than KMeans**.

### ****DBSCAN****

* **Noise Points**: 305 → These were marked as outliers (-1 label).
* **Silhouette Score**: 0.1808 → Weak cluster separation.
* **Davies-Bouldin Index**: 1.6532 → Higher = worse cluster quality.

### ****Conclusion:****

* **Agglomerative Clustering** slightly outperforms KMeans.
* **DBSCAN** is less effective here, likely due to high-dimensional data.

| Model | Silhouette Score | Davies-Bouldin | Notes |
| --- | --- | --- | --- |
| KMeans | 0.3366 | 1.0070 | Good performance overall |
| Agglomerative | 0.3339 | 0.9877 | Best Davies-Bouldin score |
| DBSCAN | 0.1808 | 1.6532 | Weak clustering, 305 noise pts |

### Best Model: Agglomerative Clustering

* Similar Silhouette Score to KMeans.
* **Best Davies-Bouldin Index** (lower = better compactness/separation).
* More **stable structure** in high-dimensional data.