BA820 - M4: Refinement

Team B02

Akbar Wibowo

1. Refined Problem Statement & Focus

This milestone investigates the question: "Are IMDb ratings mostly consistent, or are there clear high and low rated episodes or seasons?" The aim is to see whether audience reception of *Alone* is fairly uniform across the series or whether episodes and seasons naturally separate into distinct reception groups that matter for content performance and promotion decisions. IMDb rating is treated as the main reception measure, while vote count and viewership are used to add context about engagement and how reliable or comparable ratings may be.

The question itself did not change, but the focus shifted in Milestone 4 from finding clusters to testing how stable they are. Refinement experiments compared k equals 3 and k equals 5 to balance simple buckets versus more detailed tiers, then tested whether adding viewership changes cluster membership and whether missing viewership handling affects results (complete case versus imputation). These checks supported the idea that ratings contain real structure beyond random noise, while also showing that viewership adds an exposure dimension that can shift membership more at finer k. At the same time, complete case and imputed viewer versions were highly consistent on overlapping episodes, and hierarchical clustering aligned closely with KMeans, especially at k equals 5, which strengthens confidence that the refined segmentation is not dependent on a single method.

2. EDA & Preprocessing: Updates

Earlier EDA and preprocessing established the baseline structure of the IMDb data used for reception segmentation. Analysis was restricted to the U.S. version of the episodes dataset for consistency, and `air_date` was parsed to support chronological checks when needed. Data integrity checks confirmed no duplicate episode identifiers at the season and episode level. Ratings analysis focused on episodes with observed IMDb ratings, since the business question concerns actual audience feedback; this retained 93 episodes with non missing `imdb_rating`. Vote counts were retained alongside ratings because they provide context on rating reliability and engagement, and earlier checks flagged that a small number of episodes have unusually low vote volume, meaning extreme ratings should be interpreted cautiously rather than treated as equally stable across all episodes.

Milestone 4 did not require major new cleaning, but it did introduce more explicit dataset definitions to support refinement experiments. Instead of using a single filtered dataframe for all steps, three analysis ready versions were constructed: a baseline dataset requiring `imdb_rating` and `n_ratings`, a complete case viewership dataset requiring `imdb_rating`, `n_ratings`, and `viewers`, and an imputed viewership dataset that fills missing `viewers` with the median to retain the full baseline sample. This change was necessary because viewership is missing for a nontrivial subset of episodes and is concentrated in later seasons, so comparing complete case and imputed approaches helps separate feature sensitivity from missingness effects. In addition, standardization was applied consistently across feature sets prior to clustering to ensure distance based methods were not dominated by scale differences between ratings, votes, and viewers.

3. Analysis & Experiments

This milestone continues the same question from earlier work: whether IMDb ratings for Alone are broadly consistent, or whether episodes and seasons separate into meaningful reception groups. In Milestone 2 I used clustering on rating and vote count to create reception buckets. In Milestone 4, the focus shifts from discovery to refinement and validation. Instead of relying on one clustering result, I tested whether the segmentation remains stable under reasonable modeling choices, feature choices, and missing data handling. KMeans is the main method because it directly produces interpretable "buckets" in the feature space and makes it easy to compare variants. Hierarchical clustering is added as a method level robustness check.

3.1 Baseline segmentation using rating and vote count (X1)

I first used the baseline feature set X1 with imdb_rating and n_ratings because it stays closest to the business question: audience reception plus the context of how much participation supports each rating. To refine model selection, I compared inertia and silhouette across k values. The inertia curve drops steeply from k equals 2 through around k equals 5 and then flattens, suggesting diminishing returns after that point (Appendix Figure A1). Silhouette values improve around k equals 5 and then rise more gradually, which supports k equals 5 if the goal is separation, while k equals 3 remains reasonable if the goal is simpler interpretation (Appendix Figure A2). Because this milestone is explicitly a refinement stage, I carried both k equals 3 and k equals 5 forward rather than choosing a single k early.

The fitted models show that k equals 3 yields three interpretable clusters, while k equals 5 splits the space into finer tiers (Appendix Table B2). The adjusted rand index between the two segmentations is not close to 1, meaning k equals 5 is not just relabeling k equals 3. It meaningfully reshuffles some episodes, which is expected when moving from coarse buckets to more detailed tiers. Cluster profile summaries show the interpretation clearly. Under k equals 3, the structure separates into a lower rated group, a higher rated group, and a distinct high vote bucket, suggesting that vote volume behaves like an engagement or stability signal that is not identical to rating level (Appendix Table B3). Under k equals 5, the same logic remains but becomes more granular: high rated content splits into multiple tiers and the lower rated region becomes more clearly isolated (Appendix Table B4). The scatter visual of vote count versus rating supports this interpretation by showing clusters occupying different regions, especially the high vote region (Appendix Figures A3 and A4).

To connect episode clusters back to season level patterns, I used within season distributions. Season by cluster tables show that seasons are not evenly mixed across buckets. Several seasons concentrate heavily into one or two clusters, supporting the idea that reception differences are structured across seasons rather than being purely random episode noise (Appendix Tables B5 and B6). A cross tab showing how k equals 5 splits k equals 3 clarifies what “refinement” means in practice: the refined solution mostly subdivides the broad buckets rather than producing a totally different narrative (Appendix Table B7).

3.2 Feature sensitivity check using viewership (X2 and X3)

A key integration insight from Milestone 3 was that reception signals and exposure signals can get mixed. Ratings and votes represent evaluation and participation, while viewers captures reach. Adding viewers could create clusters that represent popularity rather than quality. To test sensitivity to this feature choice, I repeated clustering using viewership in two ways: X2 complete case, keeping only episodes with non missing viewers, and X3 imputed, filling missing viewers with the median to retain the baseline sample size.

When viewership is added, cluster profiles show that exposure becomes a meaningful differentiator. Some clusters now separate more on viewership levels even when rating differences are modest, reinforcing the point that popularity and reception are related but not identical (Appendix Table B9 and Appendix Table B11). Agreement metrics quantify how much cluster membership changes. X2 compared to X1 shows moderate ARI values, indicating that adding viewership shifts assignments, especially under k equals 5 where finer tiers are more sensitive to feature space changes (Appendix Table B8). This was one of the main refinement findings: the baseline reception segmentation is not completely feature invariant, so interpretation depends on whether the stakeholder question is closer to satisfaction among raters (ratings plus votes) or reach plus reception (ratings, votes, viewers).

The missingness handling result was more reassuring. X2 and X3 agree extremely strongly on overlapping episodes, meaning the segmentation is not being driven primarily by whether viewers are complete case versus imputed (Appendix Table B10). This suggests the main conceptual difference is whether viewers are included at all, not the specific missingness strategy used here. Visual diagnostics support these conclusions. Pairplots and 3D cluster visuals show that clusters occupy different regions in the combined space of ratings, votes, and viewers, and that viewership adds a clear exposure axis that does not perfectly align with rating (Appendix Figures A5 through A12). These visuals helped confirm that the “viewership model” is answering a slightly different question than the baseline model.

3.3 Method robustness using hierarchical clustering (X1)

To check whether the segmentation depends on KMeans specifically, I ran Ward hierarchical clustering on the standardized X1 features. The dendrogram provides a multi scale view of structure and suggests visible grouping patterns rather than a completely smooth, noise driven tree (Appendix Figure A13). Cutting the dendrogram to k equals 3 and k equals 5 produces reasonable cluster sizes and interpretable cluster summaries that align with the same overall segmentation logic seen with KMeans (Appendix Tables B12 and B15). Agreement with KMeans is high, especially at k equals 5. ARI values and cross tabs show that many clusters map closely across methods, supporting the claim that the refined bucket structure is robust to the choice of clustering algorithm (Appendix Tables B13 and B14). Across both k values, hierarchical clustering supports the same segmentation logic as KMeans, and the especially high agreement at k equals 5 suggests the refined bucket structure is robust to the choice of clustering method.

3.4 What worked, what did not, and what I learned

Several refinement steps worked well. Carrying both k equals 3 and k equals 5 forward made the analysis more honest and more useful: k equals 3 gives simple buckets, while k equals 5 adds tier level detail without changing the overall story. The viewership sensitivity tests were also valuable because they showed that adding reach shifts membership in meaningful ways, which prevents overinterpreting rating only clusters as a proxy for popularity. Finally, hierarchical clustering provided a strong robustness check and reduced the risk that the segmentation is a KMeans artifact.

The main limitation is interpretability as k increases. Even if k equals 5 separates better, it takes more explanation and relies more heavily on appendix tables and visuals. Another challenge is that viewership is missing for a nontrivial subset of episodes and is concentrated in later seasons, so complete case filtering could bias the viewership based segmentation. The imputation comparison helped address this concern, but it still remains a simplification rather than a definitive missing data solution. I also considered applying text analysis methods from class, but the episode level dataset does not contain strong text fields that plausibly explain IMDb ratings at scale. For that reason, I prioritized deeper robustness and sensitivity checks on the numeric reception segmentation rather than forcing a text pipeline that would be weakly connected to the actual question.

4. Findings & Interpretations

IMDb ratings for *Alone* are generally strong, but they are not perfectly uniform across the series. Even though most episodes fall in a fairly tight rating band, the clustering results show that episodes still separate into recognizable “buckets” rather than forming one single, consistent mass. In the baseline view using ratings and vote counts, a clear pattern emerges: there are episodes that tend to be rated higher, episodes that tend to sit in a lower tier, and a distinct group that stands out mainly because it attracts far more votes than typical (see Figures A3 and A4, and Tables B3 and B4). This matters because it suggests audience reception has more than one dimension. Rating level captures satisfaction, but vote volume captures how strongly viewers engage with the episode as something worth rating, which is also a useful signal for decision makers.

A second key insight is that season-level patterns are not random. When episodes are grouped into buckets, some seasons concentrate heavily in the same bucket, rather than spreading evenly across all groups (see Tables B5 and B6). In practice, that means there are production cycles where the show delivers a more consistently strong viewer experience, and other periods where episodes are more likely to land in the mid or lower tier. For stakeholders, this has direct relevance. Programming and promotion teams can highlight seasons that cluster more consistently in the higher reception tier, while content strategy teams can treat lower-tier seasons as diagnostic periods for understanding what might have changed in pacing, casting, editing, or narrative structure.

Adding viewership strengthens the interpretation because it separates “well liked” from “widely seen.” When viewership is included, the clusters become partly organized around exposure, meaning some episodes group together because they reached more people, even if their ratings are not uniquely higher (see Figures A9 through A12, and Tables B9 through B11). This reinforces a practical distinction: reach and satisfaction are related, but they are not the same thing. Some episodes are widely watched without being the best reviewed, and some are very well reviewed without being the biggest reach events. For business use, this supports two different decisions. If the goal is to maximize audience growth, viewership-heavy clusters matter most. If the goal is to maximize satisfaction and long-term series reputation, higher-rated clusters matter more even if their reach is not the highest.

The refinement tests also increase confidence in the findings. Cluster structure remains very similar when missing viewership is handled in different ways, which suggests that the viewership-related patterns are not being driven by a small subset of missing data (see Table B10). In addition, hierarchical clustering largely supports the same segmentation logic as KMeans, especially under the refined k equals 5 setting (see Figure A13 and Tables B13 through B15). In plain terms, the main reception story is not fragile: it does not disappear when the analysis is rerun with reasonable alternatives. Overall, the results support the conclusion that IMDb reception for *Alone* is broadly positive but meaningfully segmented. These segments provide a practical way to identify standout content, weaker periods, and highly engaging episodes, which can guide promotion choices and inform longer-term content planning.

Appendix

Shared GitHub Repository

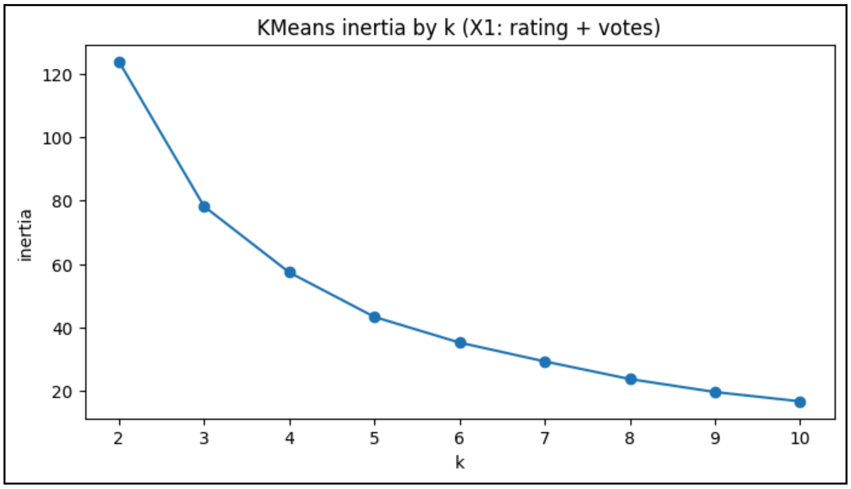
https://github.com/MikaIsmayilov/B02_Alone_TVShow_Unsupervised_Project/tree/main/M4_Refinement/Akbar

The files in the specific GitHub Repository folder is this report and the .ipynb file containing the code.
They are titled 'akbaraw2_BA820_M4Report.pdf' and 'akbaraw2_BA820_M4Notebook.ipynb'.

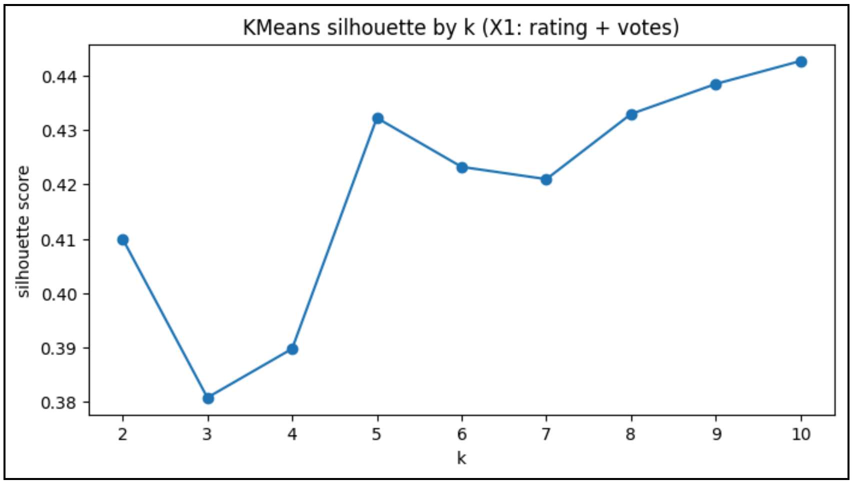
Supplemental Material (Highly Recommended)

A. Figures:

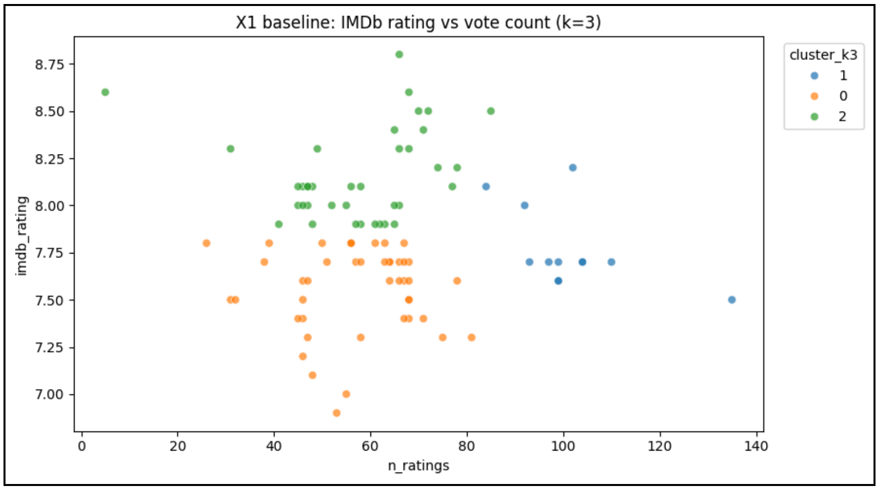
- **Figure A1, KMeans inertia vs k on X1**



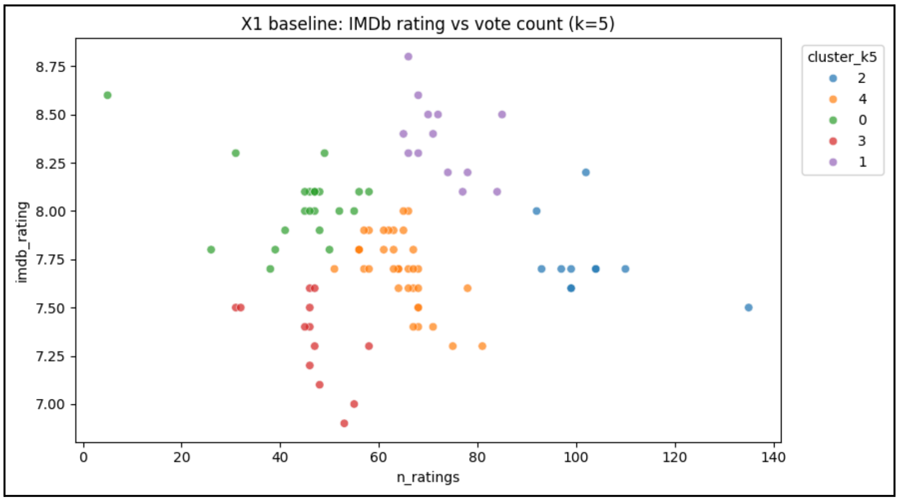
- **Figure A2, KMeans silhouette vs k on X1**



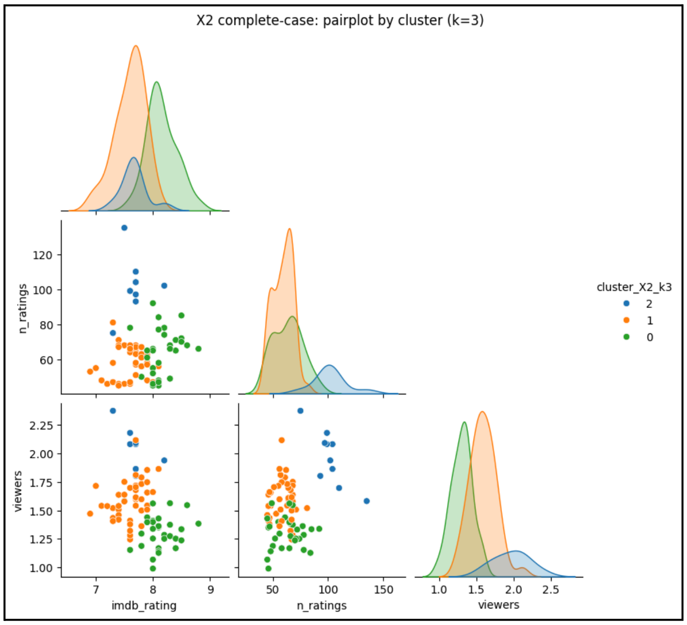
- **Figure A3, X1 scatter n_ratings vs imdb_rating colored by k equals 3**



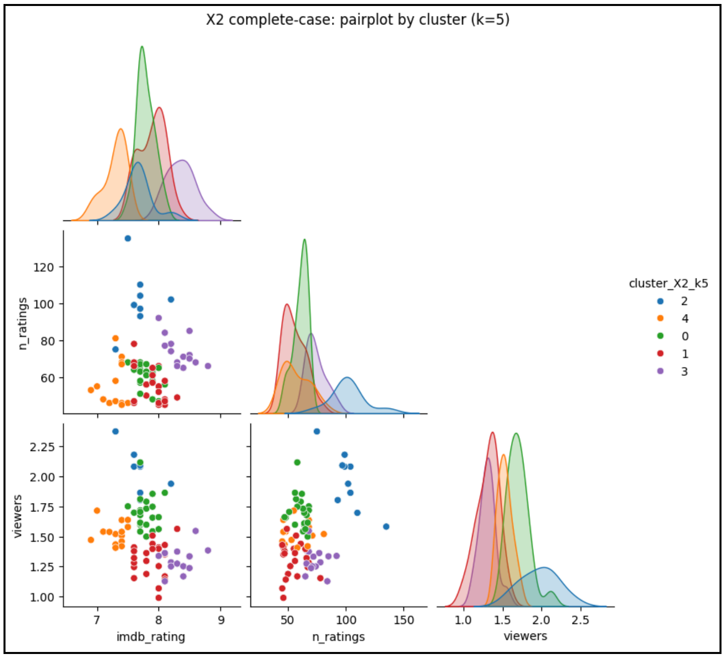
- **Figure A4, X1 scatter n_ratings vs imdb_rating colored by k equals 5**



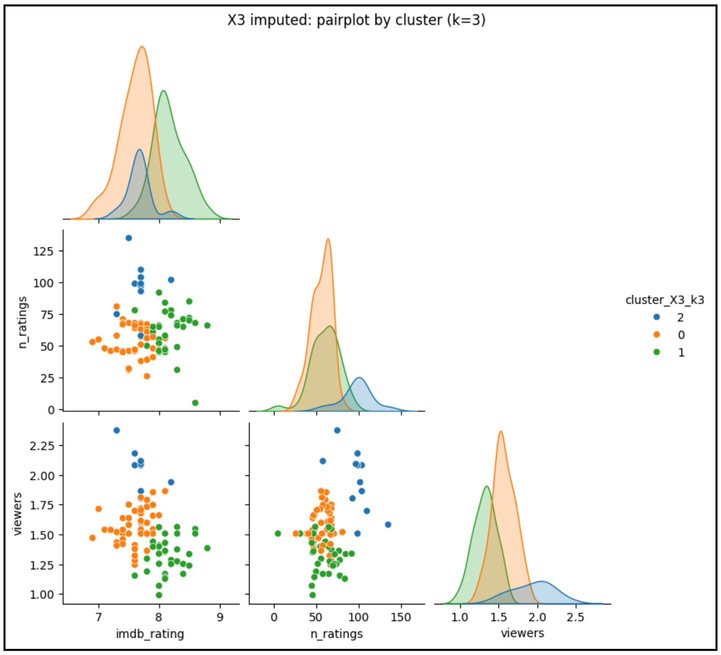
- **Figure A5, X2 pairplot k equals 3**



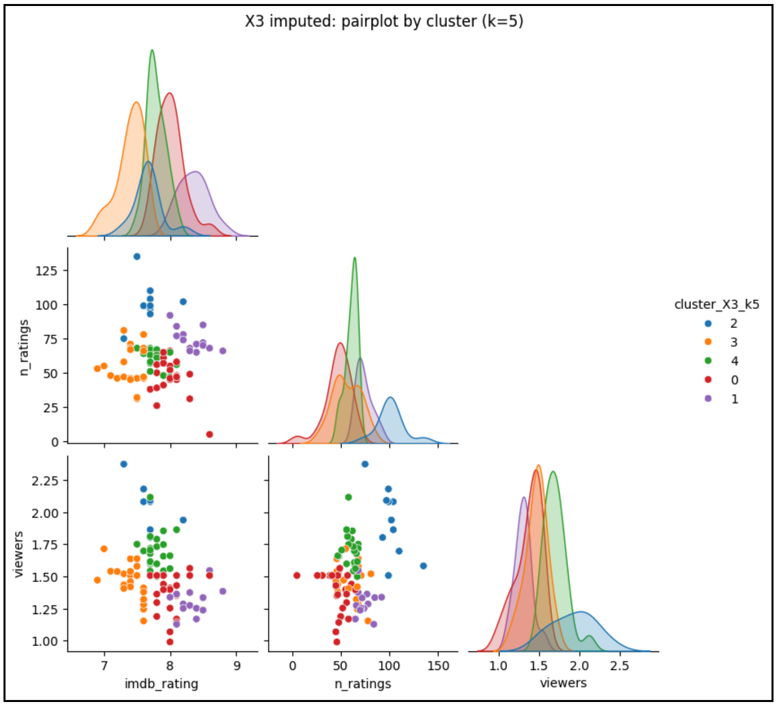
- **Figure A6, X2 pairplot k equals 5**



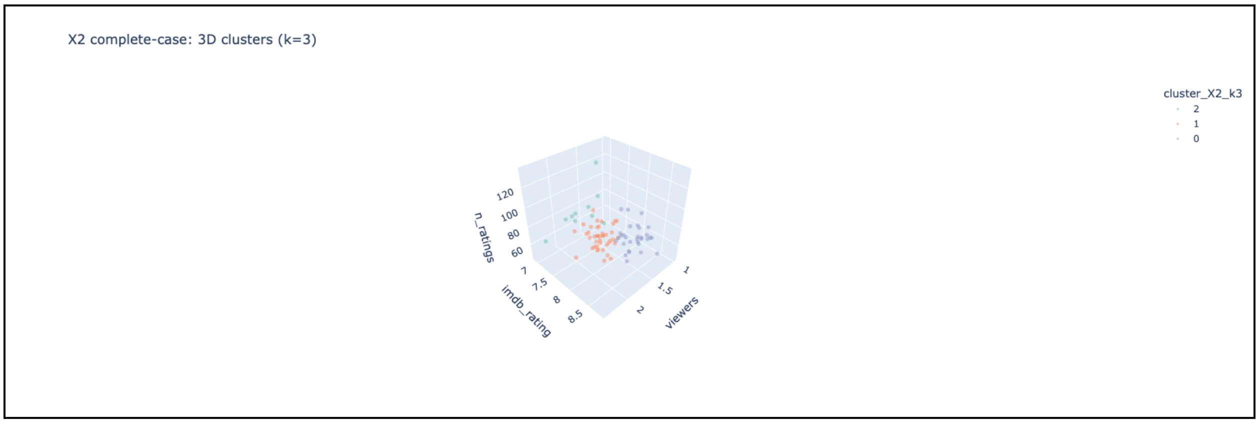
- **Figure A7, X3 pairplot k equals 3**



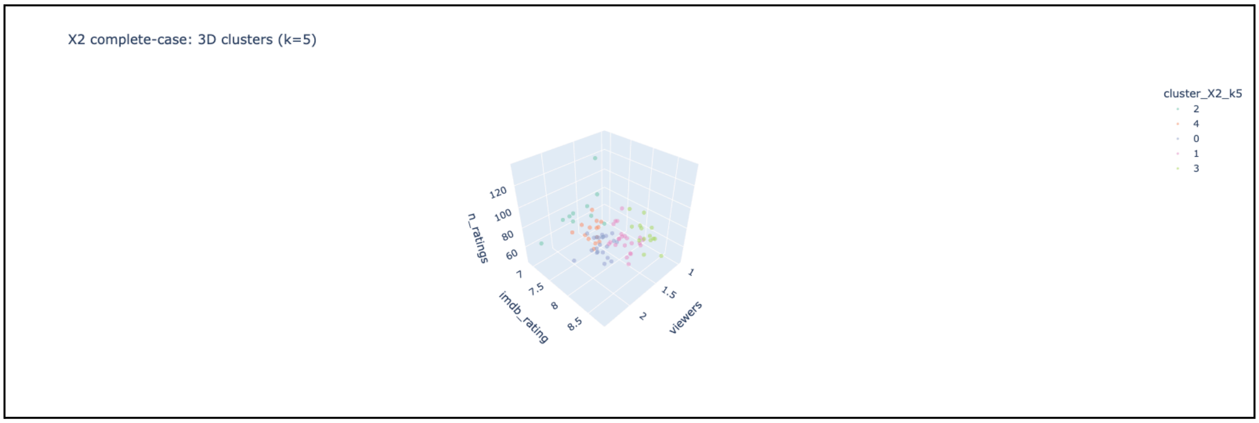
- **Figure A8, X3 pairplot k equals 5**



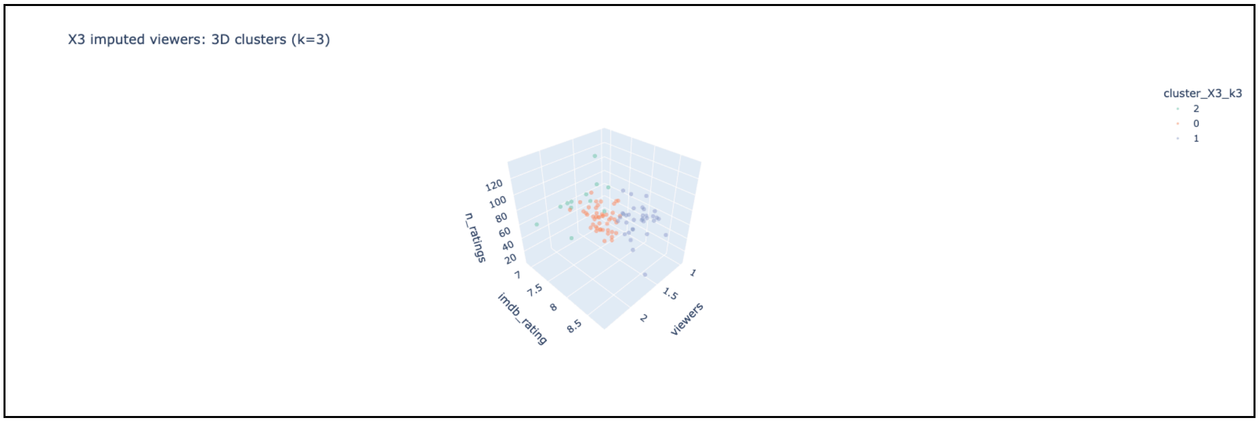
- **Figure A9, X2 3D scatter k equals 3**



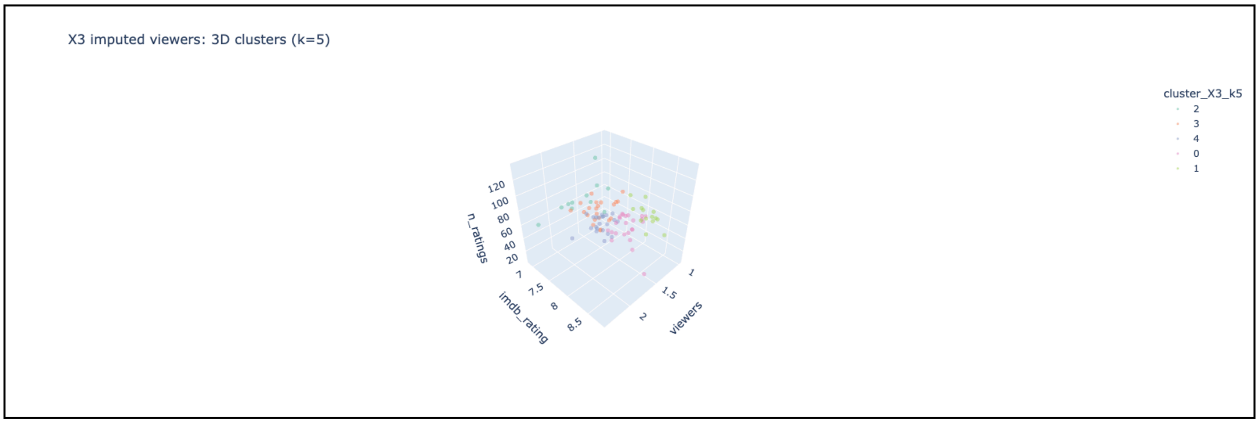
- **Figure A10, X2 3D scatter k equals 5**



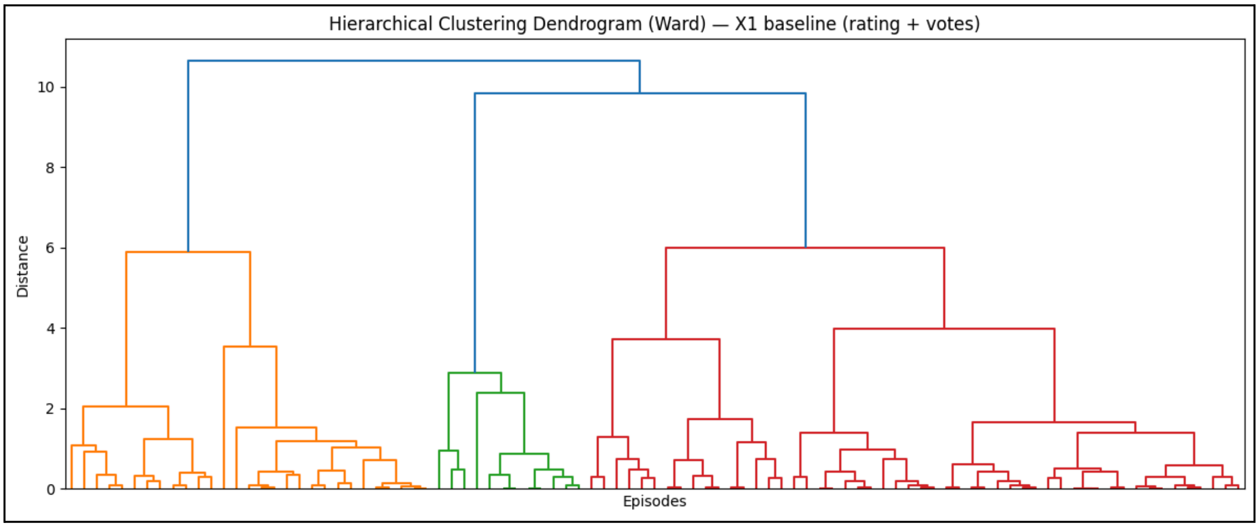
- **Figure A11, X3 3D scatter k equals 3**



- **Figure A12, X3 3D scatter k equals 5**

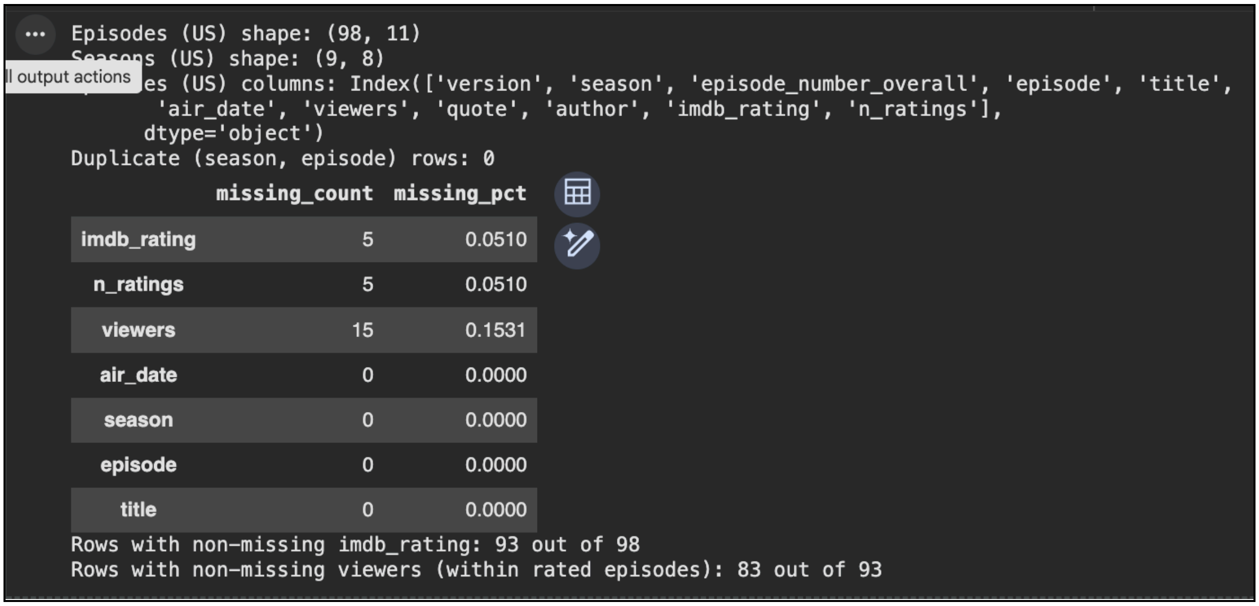


- **Figure A13, Ward dendrogram on X1**



B. Tables:

- **Table B1, missingness summary and dataset sizes**

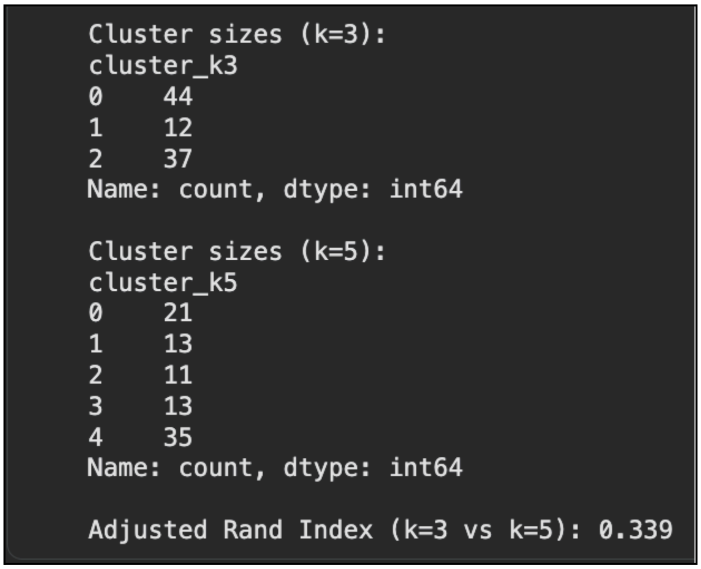


```
Episodes (US) shape: (98, 11)
Seasons (US) shape: (9, 8)
      es (US) columns: Index(['version', 'season', 'episode_number_overall', 'episode', 'title',
       'air_date', 'viewers', 'quote', 'author', 'imdb_rating', 'n_ratings'],
      dtype='object')
Duplicate (season, episode) rows: 0
```

	missing_count	missing_pct
imdb_rating	5	0.0510
n_ratings	5	0.0510
viewers	15	0.1531
air_date	0	0.0000
season	0	0.0000
episode	0	0.0000
title	0	0.0000

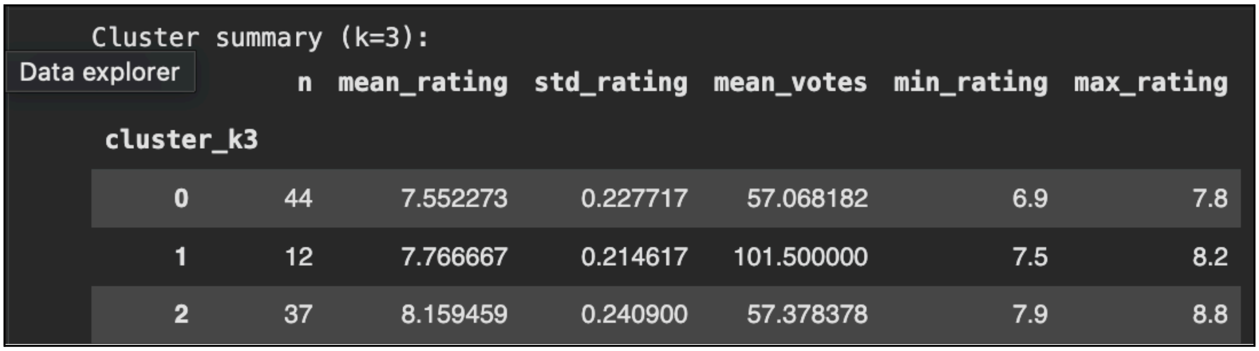
```
Rows with non-missing imdb_rating: 93 out of 98
Rows with non-missing viewers (within rated episodes): 83 out of 93
```

- **Table B2, X1 cluster sizes for k equals 3 and k equals 5 plus ARI between them**



```
Cluster sizes (k=3):
cluster_k3
0    44
1    12
2    37
Name: count, dtype: int64

Cluster sizes (k=5):
cluster_k5
0    21
1    13
2    11
3    13
4    35
Name: count, dtype: int64

Adjusted Rand Index (k=3 vs k=5): 0.339
```

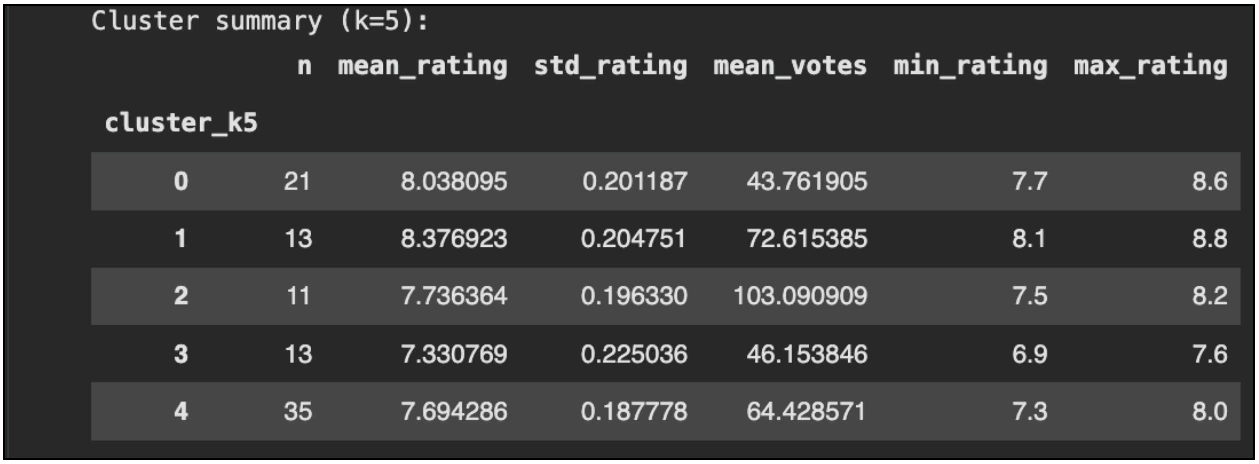
- **Table B3, X1 cluster profile summary for k equals 3**



Cluster summary (k=3):

	n	mean_rating	std_rating	mean_votes	min_rating	max_rating
cluster_k3						
0	44	7.552273	0.227717	57.068182	6.9	7.8
1	12	7.766667	0.214617	101.500000	7.5	8.2
2	37	8.159459	0.240900	57.378378	7.9	8.8

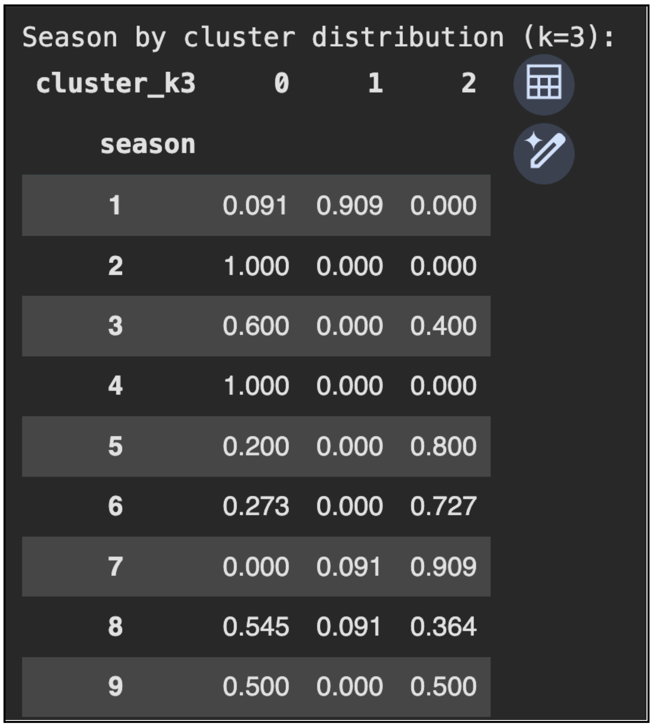
- **Table B4, X1 cluster profile summary for k equals 5**



Cluster summary (k=5):

cluster_k5	n	mean_rating	std_rating	mean_votes	min_rating	max_rating
0	21	8.038095	0.201187	43.761905	7.7	8.6
1	13	8.376923	0.204751	72.615385	8.1	8.8
2	11	7.736364	0.196330	103.090909	7.5	8.2
3	13	7.330769	0.225036	46.153846	6.9	7.6
4	35	7.694286	0.187778	64.428571	7.3	8.0

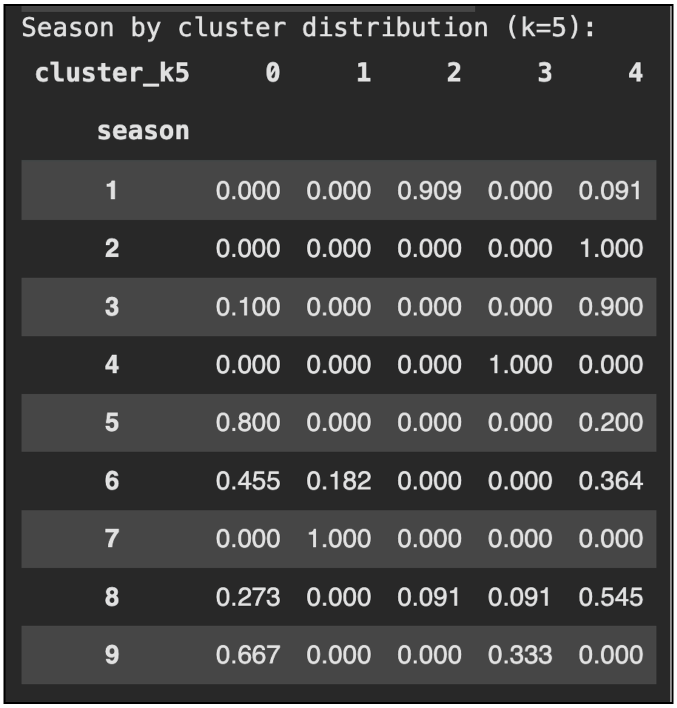
- **Table B5, season by cluster distribution for X1, k equals 3 (row normalized)**



Season by cluster distribution (k=3):

cluster_k3 season	0	1	2
1	0.091	0.909	0.000
2	1.000	0.000	0.000
3	0.600	0.000	0.400
4	1.000	0.000	0.000
5	0.200	0.000	0.800
6	0.273	0.000	0.727
7	0.000	0.091	0.909
8	0.545	0.091	0.364
9	0.500	0.000	0.500

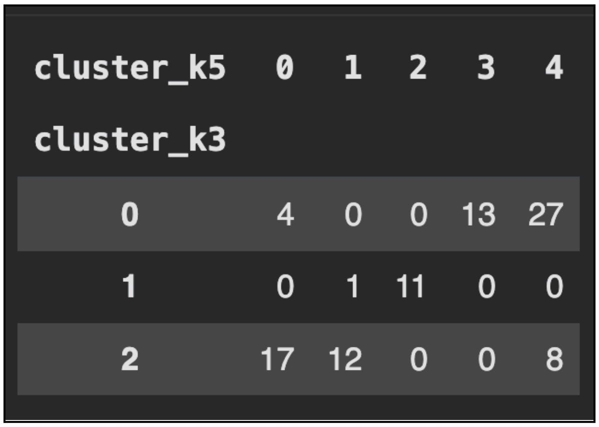
- **Table B6, season by cluster distribution for X1, k equals 5 (row normalized)**



Season by cluster distribution (k=5):

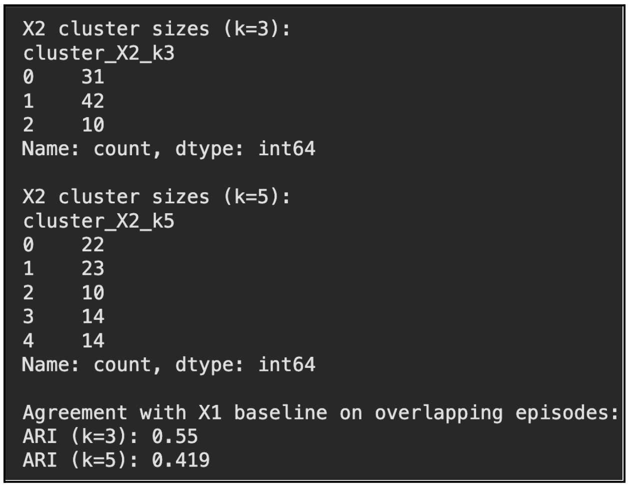
cluster_k5	0	1	2	3	4
season					
1	0.000	0.000	0.909	0.000	0.091
2	0.000	0.000	0.000	0.000	1.000
3	0.100	0.000	0.000	0.000	0.900
4	0.000	0.000	0.000	1.000	0.000
5	0.800	0.000	0.000	0.000	0.200
6	0.455	0.182	0.000	0.000	0.364
7	0.000	1.000	0.000	0.000	0.000
8	0.273	0.000	0.091	0.091	0.545
9	0.667	0.000	0.000	0.333	0.000

- **Table B7, crosstab showing how k equals 5 splits k equals 3 (X1)**



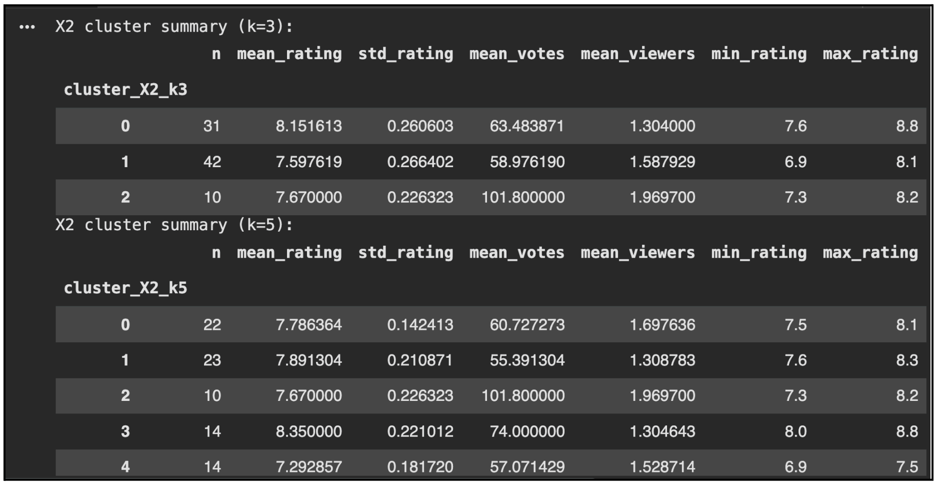
cluster_k5	0	1	2	3	4
cluster_k3					
0	4	0	0	13	27
1	0	1	11	0	0
2	17	12	0	0	8

- **Table B8, agreement between X1 and X2 using ARI (k equals 3 and k equals 5)**



```
X2 cluster sizes (k=3):
cluster_X2_k3
0    31
1    42
2    10
Name: count, dtype: int64

X2 cluster sizes (k=5):
cluster_X2_k5
0    22
1    23
2    10
3    14
4    14
Name: count, dtype: int64

Agreement with X1 baseline on overlapping episodes:
ARI (k=3): 0.55
ARI (k=5): 0.419
```

- **Table B9, X2 cluster summaries and season by cluster tables (k equals 3 and k equals 5)**

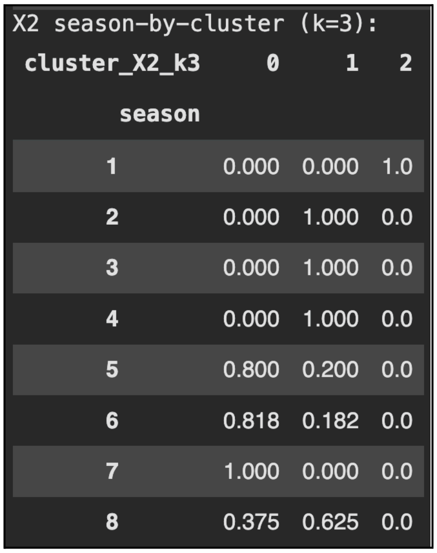


X2 cluster summary (k=3):

	n	mean_rating	std_rating	mean_votes	mean_viewers	min_rating	max_rating
cluster_X2_k3							
0	31	8.151613	0.260603	63.483871	1.304000	7.6	8.8
1	42	7.597619	0.266402	58.976190	1.587929	6.9	8.1
2	10	7.670000	0.226323	101.800000	1.969700	7.3	8.2

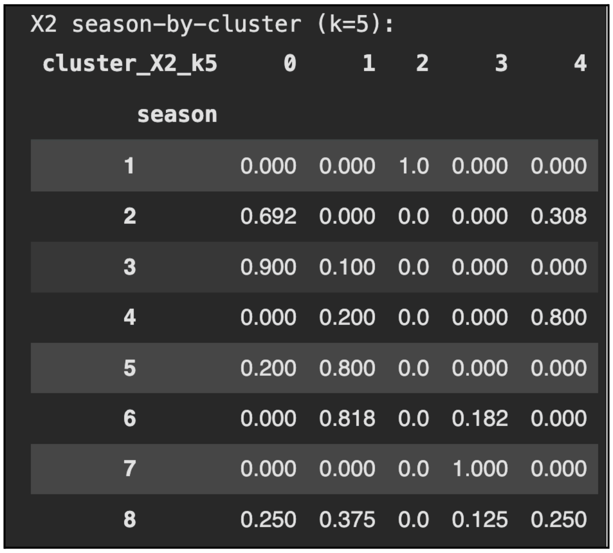
X2 cluster summary (k=5):

	n	mean_rating	std_rating	mean_votes	mean_viewers	min_rating	max_rating
cluster_X2_k5							
0	22	7.786364	0.142413	60.727273	1.697636	7.5	8.1
1	23	7.891304	0.210871	55.391304	1.308783	7.6	8.3
2	10	7.670000	0.226323	101.800000	1.969700	7.3	8.2
3	14	8.350000	0.221012	74.000000	1.304643	8.0	8.8
4	14	7.292857	0.181720	57.071429	1.528714	6.9	7.5



X2 season-by-cluster (k=3):

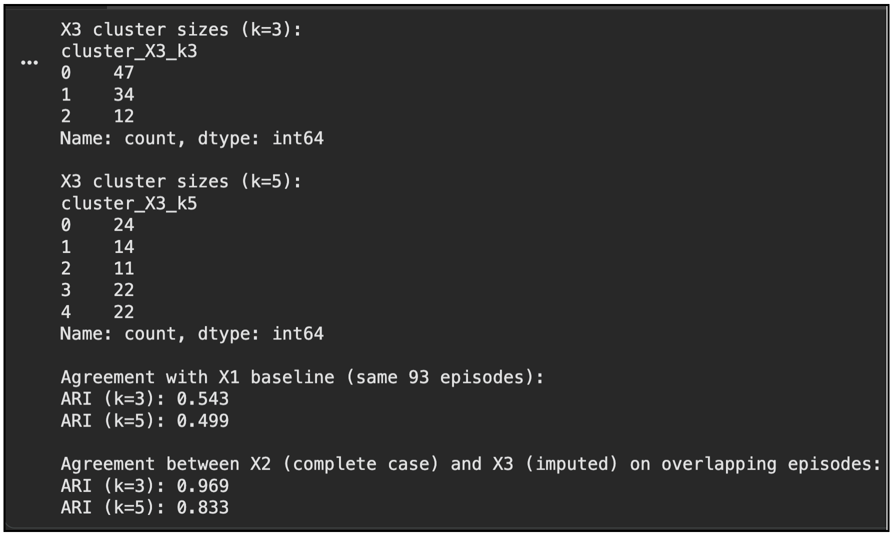
cluster_X2_k3	0	1	2
season			
1	0.000	0.000	1.0
2	0.000	1.000	0.0
3	0.000	1.000	0.0
4	0.000	1.000	0.0
5	0.800	0.200	0.0
6	0.818	0.182	0.0
7	1.000	0.000	0.0
8	0.375	0.625	0.0



X2 season-by-cluster (k=5):

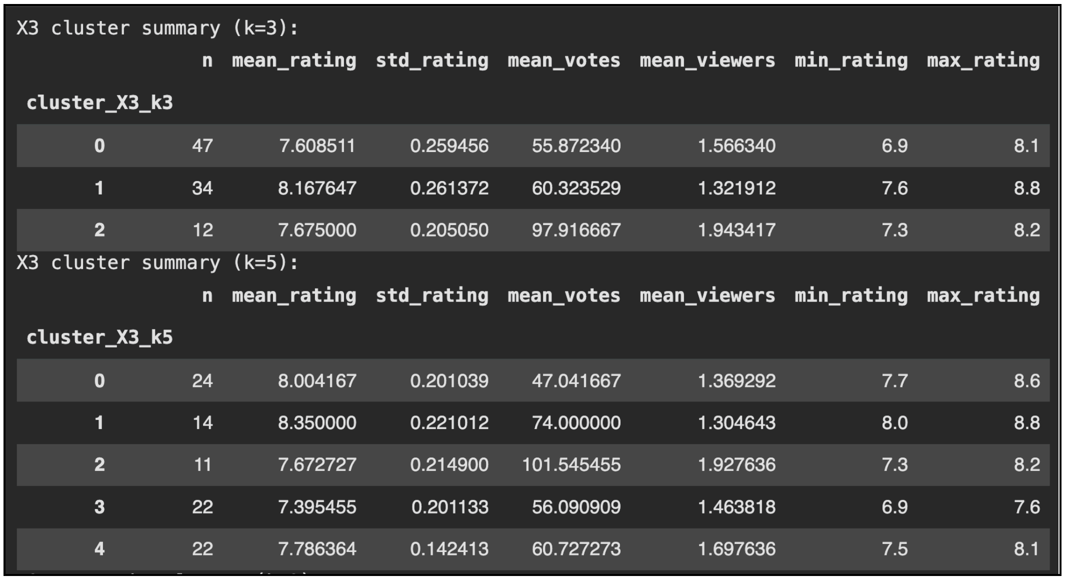
cluster_X2_k5	0	1	2	3	4
season					
1	0.000	0.000	1.0	0.000	0.000
2	0.692	0.000	0.0	0.000	0.308
3	0.900	0.100	0.0	0.000	0.000
4	0.000	0.200	0.0	0.000	0.800
5	0.200	0.800	0.0	0.000	0.000
6	0.000	0.818	0.0	0.182	0.000
7	0.000	0.000	0.0	1.000	0.000
8	0.250	0.375	0.0	0.125	0.250

- **Table B10, agreement between X1 and X3 plus agreement between X2 and X3 (k equals 3 and k equals 5)**



```
…   X3 cluster sizes (k=3):
    cluster_X3_k3
    0    47
    1    34
    2    12
    Name: count, dtype: int64

    X3 cluster sizes (k=5):
    cluster_X3_k5
    0    24
    1    14
    2    11
    3    22
    4    22
    Name: count, dtype: int64

    Agreement with X1 baseline (same 93 episodes):
    ARI (k=3): 0.543
    ARI (k=5): 0.499

    Agreement between X2 (complete case) and X3 (imputed) on overlapping episodes:
    ARI (k=3): 0.969
    ARI (k=5): 0.833
```

- **Table B11, X3 cluster summaries and season by cluster tables (k equals 3 and k equals 5)**

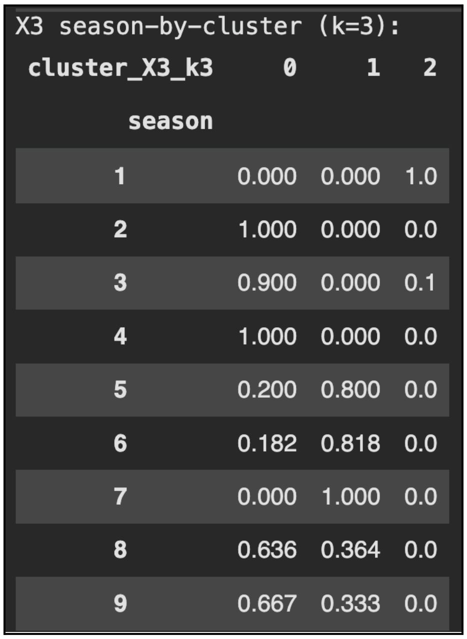


X3 cluster summary (k=3):

cluster_X3_k3	n	mean_rating	std_rating	mean_votes	mean_viewers	min_rating	max_rating
0	47	7.608511	0.259456	55.872340	1.566340	6.9	8.1
1	34	8.167647	0.261372	60.323529	1.321912	7.6	8.8
2	12	7.675000	0.205050	97.916667	1.943417	7.3	8.2

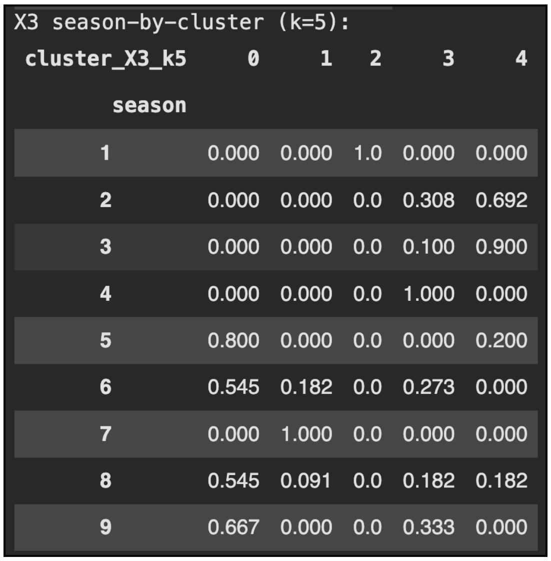
X3 cluster summary (k=5):

cluster_X3_k5	n	mean_rating	std_rating	mean_votes	mean_viewers	min_rating	max_rating
0	24	8.004167	0.201039	47.041667	1.369292	7.7	8.6
1	14	8.350000	0.221012	74.000000	1.304643	8.0	8.8
2	11	7.672727	0.214900	101.545455	1.927636	7.3	8.2
3	22	7.395455	0.201133	56.090909	1.463818	6.9	7.6
4	22	7.786364	0.142413	60.727273	1.697636	7.5	8.1



X3 season-by-cluster (k=3):

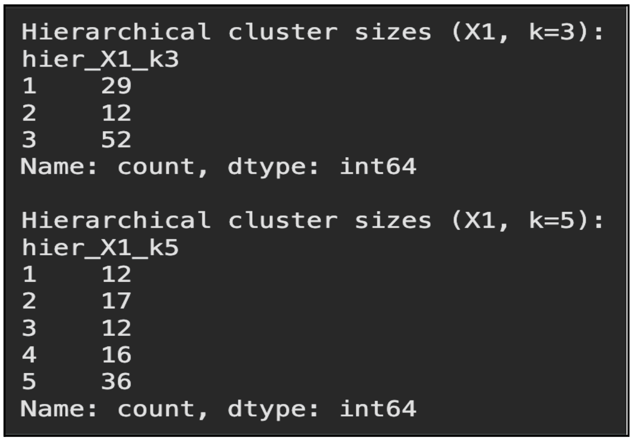
cluster_X3_k3	0	1	2
season			
1	0.000	0.000	1.0
2	1.000	0.000	0.0
3	0.900	0.000	0.1
4	1.000	0.000	0.0
5	0.200	0.800	0.0
6	0.182	0.818	0.0
7	0.000	1.000	0.0
8	0.636	0.364	0.0
9	0.667	0.333	0.0



X3 season-by-cluster (k=5):

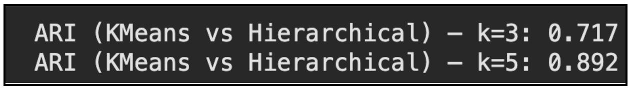
cluster_X3_k5	0	1	2	3	4
season					
1	0.000	0.000	1.0	0.000	0.000
2	0.000	0.000	0.0	0.308	0.692
3	0.000	0.000	0.0	0.100	0.900
4	0.000	0.000	0.0	1.000	0.000
5	0.800	0.000	0.0	0.000	0.200
6	0.545	0.182	0.0	0.273	0.000
7	0.000	1.000	0.0	0.000	0.000
8	0.545	0.091	0.0	0.182	0.182
9	0.667	0.000	0.0	0.333	0.000

- **Table B12, hierarchical cluster sizes for X1 (k equals 3 and k equals 5)**



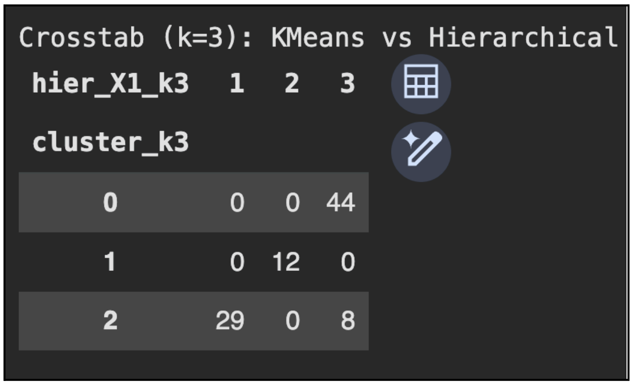
```
Hierarchical cluster sizes (X1, k=3):
hier_X1_k3
1    29
2    12
3    52
Name: count, dtype: int64

Hierarchical cluster sizes (X1, k=5):
hier_X1_k5
1    12
2    17
3    12
4    16
5    36
Name: count, dtype: int64
```

- **Table B13, ARI KMeans versus hierarchical for k equals 3 and k equals 5**



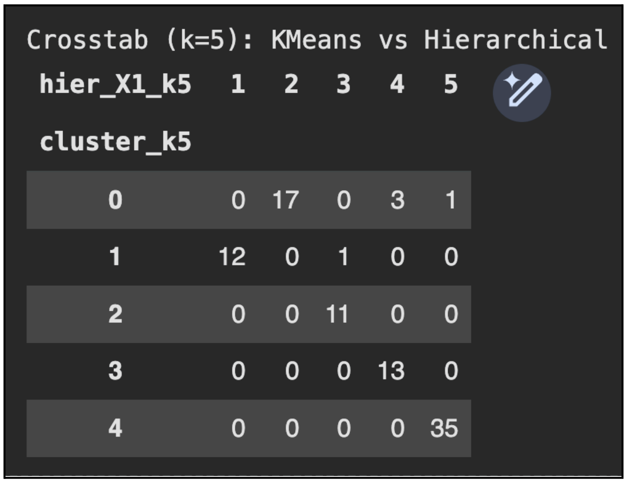
```
ARI (KMeans vs Hierarchical) – k=3: 0.717
ARI (KMeans vs Hierarchical) – k=5: 0.892
```

- **Table B14, crosstab mapping KMeans to hierarchical for k equals 3 and k equals 5**



Crosstab (k=3): KMeans vs Hierarchical

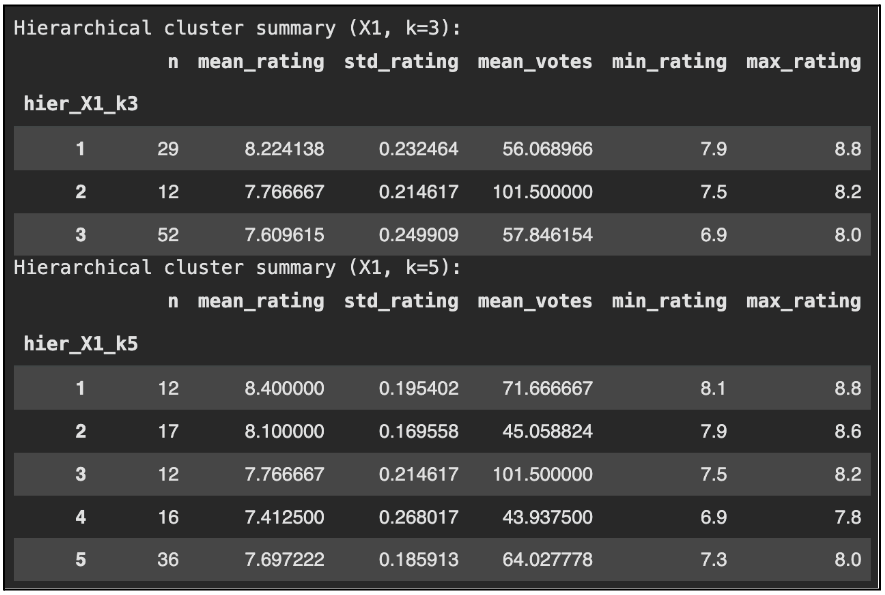
hier_X1_k3	1	2	3
cluster_k3			
0	0	0	44
1	0	12	0
2	29	0	8



Crosstab (k=5): KMeans vs Hierarchical

hier_X1_k5	1	2	3	4	5
cluster_k5					
0	0	17	0	3	1
1	12	0	1	0	0
2	0	0	11	0	0
3	0	0	0	13	0
4	0	0	0	0	35

- **Table B15, hierarchical cluster profile summaries for k equals 3 and k equals 5**



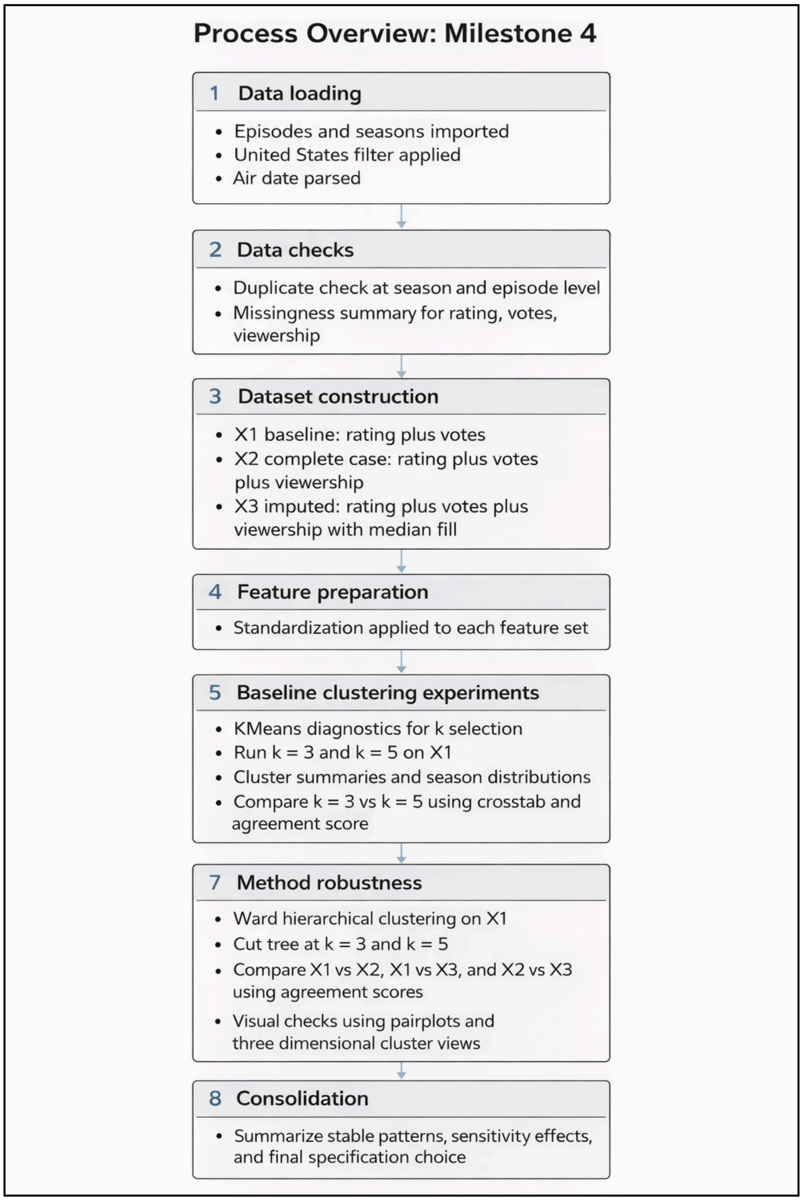
Hierarchical cluster summary (X1, k=3):

	n	mean_rating	std_rating	mean_votes	min_rating	max_rating
hier_X1_k3						
1	29	8.224138	0.232464	56.068966	7.9	8.8
2	12	7.766667	0.214617	101.500000	7.5	8.2
3	52	7.609615	0.249909	57.846154	6.9	8.0

Hierarchical cluster summary (X1, k=5):

	n	mean_rating	std_rating	mean_votes	min_rating	max_rating
hier_X1_k5						
1	12	8.400000	0.195402	71.666667	8.1	8.8
2	17	8.100000	0.169558	45.058824	7.9	8.6
3	12	7.766667	0.214617	101.500000	7.5	8.2
4	16	7.412500	0.268017	43.937500	6.9	7.8
5	36	7.697222	0.185913	64.027778	7.3	8.0

Process Overview



Process Overview: Milestone 4

1 Data loading

- Episodes and seasons imported
- United States filter applied
- Air date parsed

2 Data checks

- Duplicate check at season and episode level
- Missingness summary for rating, votes, viewership

3 Dataset construction

- X1 baseline: rating plus votes
- X2 complete case: rating plus votes plus viewership
- X3 imputed: rating plus votes plus viewership with median fill

4 Feature preparation

- Standardization applied to each feature set

5 Baseline clustering experiments

- KMeans diagnostics for k selection
- Run k = 3 and k = 5 on X1
- Cluster summaries and season distributions
- Compare k = 3 vs k = 5 using crosstab and agreement score

7 Method robustness

- Ward hierarchical clustering on X1
- Cut tree at k = 3 and k = 5
- Compare X1 vs X2, X1 vs X3, and X2 vs X3 using agreement scores
- Visual checks using pairplots and three dimensional cluster views

8 Consolidation

- Summarize stable patterns, sensitivity effects, and final specification choice

Use of Generative AI Tools

The usage of Generative AI Tools for this project milestone are divided into two methods: M4 Roadmapping and Code Formatting/Editing. The main, and only, platform used for these tools is ChatgpGPT model 5.2 Thinking.

M4 Roadmapping: https://chatgpt.com/share/699d1d2d-0a1c-8007-b248-5e68cc41dbd2

M4 Code Formatting/Editing:
https://chatgpt.com/share/699d222a-6374-8007-ba48-bc76b5e430ce