

BA820 – Project M2

Team B02

Steven Marathias

1. Refined Problem Statement & Focus (~0.5 page)

Refined question (M2): *What patterns exist in contestants' tap-out narratives? Do distinct "exit themes" show up across seasons, and can we group exits into clear buckets that producers, safety teams, and future contestants can use?*

M2 Submission Format

In Milestone 1, our team explored the Alone dataset broadly (demographics, days lasted, viewership, and item frequency). In this milestone, I narrowed my focus to the tap-out narrative text (the "reason_tapped_out" field). I made that change because the text has direct real-world value: it can help casting (who is mentally ready), safety/medical planning (common failure modes), and training (what tends to break people first).

What changed and why:

My scope shifted from "general survival advantage" to a more specific question about why contestants leave and whether those reasons cluster into repeatable themes. I expected longer, story-like explanations. After checking the data, I realized most tap-out entries are extremely short (often only a few words). That pushed me to treat this as a short-text theme discovery problem rather than a deep narrative analysis.

Assumptions that changed:

- Challenged: I assumed tap-out narratives would be detailed enough for strong topic modeling. In reality, many entries are too short to support nuanced themes.
- Validated: I assumed the text would still contain signals for a few high-level exit types (family, injury, weight loss/medical). That held up, even with short text.
- Invalidated: I assumed each contestant would contribute a unique text. Exact duplicates appear often, which affects clustering and can create "fake certainty" in results.

2. EDA & Preprocessing: Updates (~0.75 page)

M1 findings that motivated my current analysis

Two M1 results pushed me toward tap-outs. First, the show is a "scarcity + endurance" environment, so failure modes matter as much as days lasted. Second, outcomes like medical evacuation and early exits likely connect to recurring reasons (injury, family, starvation). Those observations made the tap-out text the most direct place to look for patterns.

New preprocessing and why I needed it

I focused on the survivalists table and the reason_tapped_out column. I found 94 rows total. About 10.6% of rows had missing tap-out text, and I saw many exact duplicates (23 duplicates). The raw entries were also very short: the average was about 3 to 4 words, and the maximum was only 11 words. These issues forced me to clean carefully and filter aggressively.

I also initially created a pairwise plot to look for relationships across numeric variables, but I chose not to use it in my final analysis. The plot did not make sense for this dataset because my main signal comes from short text reasons, and the few numeric fields available do not behave like continuous features with clean pairwise relationships. It also produced cluttered, unhelpful visuals given the small sample size and mixed variable types, so I moved on to text-focused methods instead.

What I changed in M2 preprocessing:

1. Text cleaning pipeline: I lowercased text, removed punctuation, collapsed extra spaces, and removed stopwords (while keeping negations like “not” and “never” when possible).
2. Short text filtering: I flagged narratives with fewer than 5 words as “short.” I filtered these out for modeling because topic models and embeddings struggle when most documents are 1 to 3 words long.
3. N grams: I used unigrams and bigrams because important phrases show up as pairs (example: “low bmi,” “missed family”).
4. Duplicate awareness: I did not delete duplicates at this stage, but I tracked them because duplicates can create repeated points that distort silhouette scores and can force KMeans into fewer distinct clusters than requested.

If I did not make these changes, the methods would mostly cluster on repeated phrases and noise, not on meaningful themes.

3. Analysis & Experiments (~1.5 page)

I applied two unsupervised approaches to detect themes in tap out reasons. Both methods aim to answer the same question: *Do exits fall into repeatable buckets, and what are those buckets?*

M2 Instructions

Method 1: Topic Modeling (NMF, with LDA as a comparison)

What this helps with:

Topic modeling helps me discover common “reason types” without labels. If the model works, it should surface interpretable themes like “injury,” “family,” “weight loss,” or “medical evacuation.”

Why it fits this data and goal:

The tap-out field is text. Topic models are designed to find patterns in word co-occurrence across documents. Even if documents are short, I expected a few strong repeated phrases.

What I tried (and parameter choices):

- Vectorization: TF-IDF with unigrams and bigrams (to capture phrases like “low bmi”).
- Model: NMF with $K = 6$ topics (first pass), using stable initialization and more iterations to converge.
- Variant / comparison: LDA with $K = 6$ and $K = 8$ using a count vectorizer, because LDA is a common alternative.

What worked / what did not / what surprised me:

- What worked: NMF surfaced a clear “weight loss / low BMI” theme and a “missed family / guilt” theme. Those topics showed up consistently and matched real exit types.
- What did not work: Many NMF topics looked repetitive and overlapped heavily. That happened because the text is extremely short, so the same few phrases dominate multiple topics.
- LDA was weaker: LDA produced multiple near-duplicate topics that repeated the same keywords (“worth it,” “low bmi,” “lost weight”). With short documents, LDA struggled to separate themes cleanly.
- Surprising result: I expected a wider spread of language, but the dataset reads more like a coded summary than a narrative. That limits how “deep” topic modeling can go.

What I learned:

Topic modeling can still work here, but only for high-level buckets, not nuanced themes. The method becomes much more about repeated phrases than storytelling language.

Method 2: Embeddings + Clustering (TF-IDF → SVD/LSA → KMeans)

What this helps with:

This method groups contestants based on overall text similarity, even when exact words differ. It is useful when topic models feel too rigid or when you want clusters that behave like “segments.”

Why it fits this data and goal:

Short text often benefits from a pipeline that compresses text into a lower-dimensional space (LSA/SVD), then clusters in that space. It also gives me tools to check cluster quality using silhouette-based evaluation.

What I tried (and parameter choices):

- Vectorization: TF-IDF with unigrams + bigrams, with mild filtering so rare terms do not explode the feature space.
- Dimensionality reduction: SVD (LSA) to up to 100 dimensions (capped by vocabulary size).
- Clustering: KMeans with multiple K values: $K = 4, 5, 6, 7, 8, 10$.
- Model selection: I selected the final K by choosing the value with the highest silhouette score, and then generated a SilhouetteVisualizer (Yellowbrick) plot for the final model as

a visual validation check.

What worked / what did not / what surprised me:

- What worked: The silhouette-based approach favored $K = 5$, and the resulting clusters were easy to label using representative examples. The clusters aligned with real exit categories:
 - Cluster (family): “missed family / felt guilty / left them behind”
 - Cluster (weight loss): “low bmi / lost much weight”
 - Cluster (injury): “broken teeth / torn meniscus / MCL”
 - Cluster (joint decision / not worth it): “jointly decided... wasn’t worth it”
 - Cluster (life event / closure): “mother’s cancer” and “felt content what he done”
- What did not work: Cluster sizes were tiny after filtering (many clusters had only 2–4 documents). That limits how confident I can be that I found stable segments, rather than small pockets driven by duplicates.
- Technical challenge: KMeans warned that it found fewer distinct clusters than requested at higher K values. That likely happened because duplicates create identical points, so the model cannot form truly distinct clusters.
- Surprising result: Even with short text, clustering produced more “usable” buckets than topic modeling. It gave me cleaner groups that map well to business interpretation.

What I learned:

For this dataset, embeddings + clustering feels more practical than classic topic modeling. The clusters behave like simple “exit segments,” which is what stakeholders would want.

4. Findings & Interpretations (~0.75 page)

At this stage, I can support one main takeaway: tap out reasons clustered into a few repeatable exit themes, but the dataset supports only coarse themes, not deep narrative insight.

Key insights:

1. Weight loss / low BMI shows up as a dominant exit theme. The language repeats strongly (“low bmi,” “lost much weight”). This suggests that physical decline is not random; it is a common failure mode that may be predictable with better risk screening or training.

2. Family / guilt exits form a clear emotional category. “Missed family” and “felt guilty” appear together and cluster cleanly. This can inform casting and pre-show preparation (mental readiness, family situation, support planning).
3. Injury exists as a separate bucket. Examples like knee damage and dental issues stand out as different from weight loss or family reasons. That can help safety teams plan medical checks and prevention guidance.
4. Some exits look like “strategic” or “joint decision” exits. Phrases like “wasn’t worth it” and “jointly decided” suggest a different mindset than collapse or injury. Producers could treat these as a distinct story type and design interview prompts around it.

Business relevance:

If producers and medical teams can tag exits into these buckets early, they can (1) balance seasons for storytelling, (2) adjust safety planning, and (3) help contestants train for the most common failure modes.

5. Next Steps (~0.25 page)

What I have not done yet:

- I have not tested whether these themes relate to days lasted, medically evacuated, season location, or gender/age differences.
- I have not improved text richness (for example, by merging with episode quotes or using longer written bios if available).

What I plan to do next:

1. Merge theme labels back to the full survivalist table and compare outcomes (days lasted, medical evacuation) across themes.
2. Reduce duplicate impact by collapsing exact duplicate narratives or weighting unique texts, then rerun clustering to see if K stays stable.
3. Try a category-level analysis using the existing “reason_category” column as a reality check, then see where my clusters match or disagree.
4. If time allows, bring in additional text (episode quotes or other fields) to make topic modeling more meaningful.

What questions remain open:

- Do “family exits” happen earlier than “injury exits”?
- Are weight loss exits more common in certain seasons or locations?
- Does the show’s coded wording hide detail that exists elsewhere?

My current findings justify these next steps because I now have a working draft of theme labels, but I need to validate whether they connect to measurable outcomes.

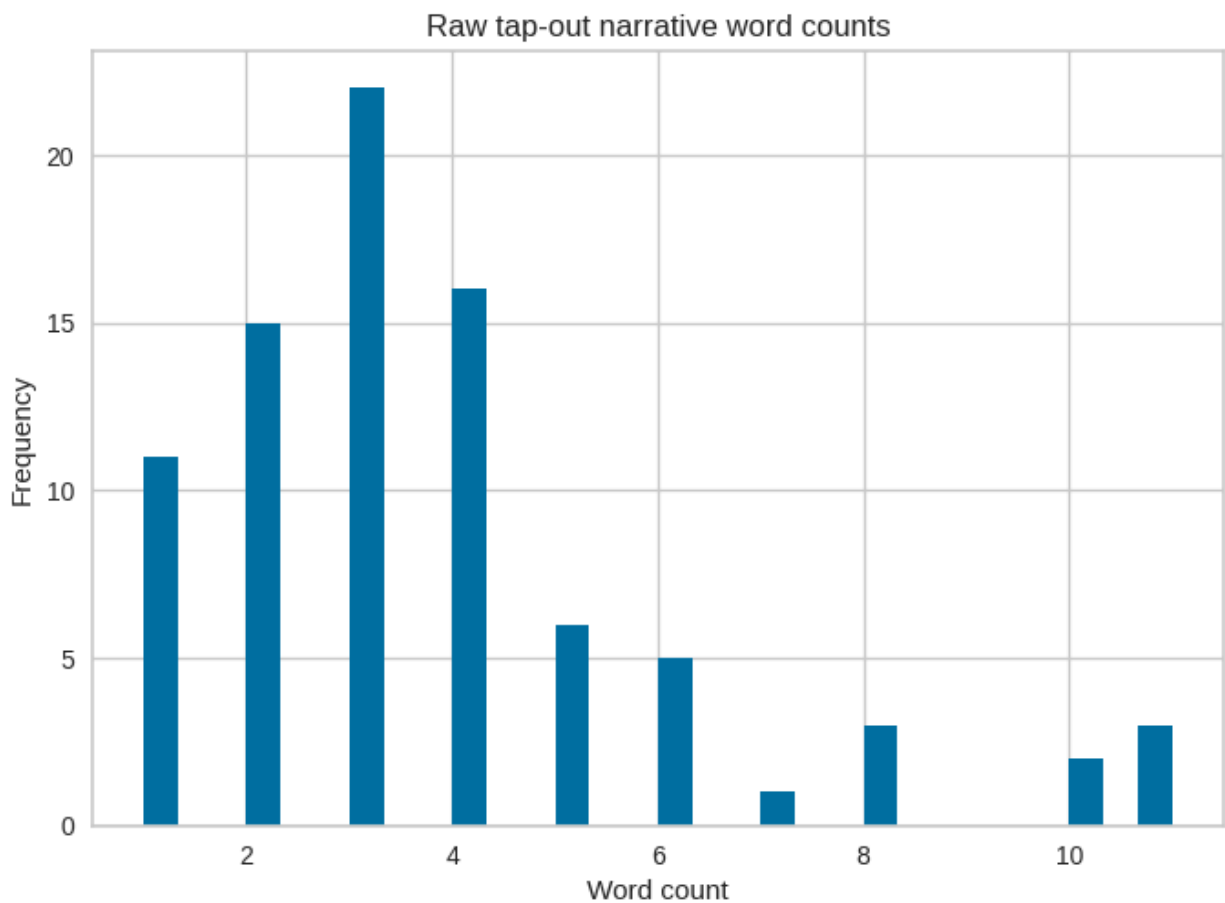
Appendix

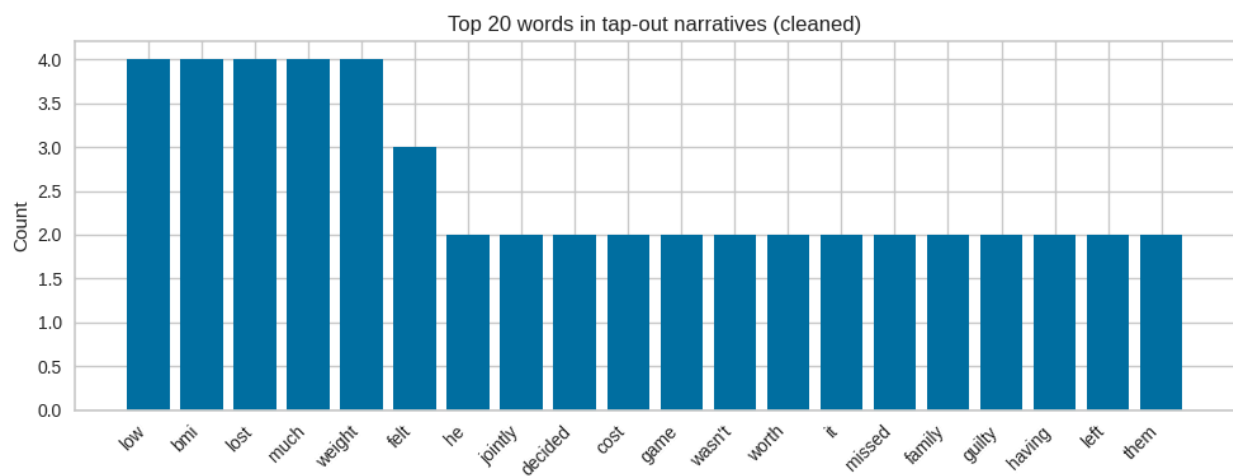
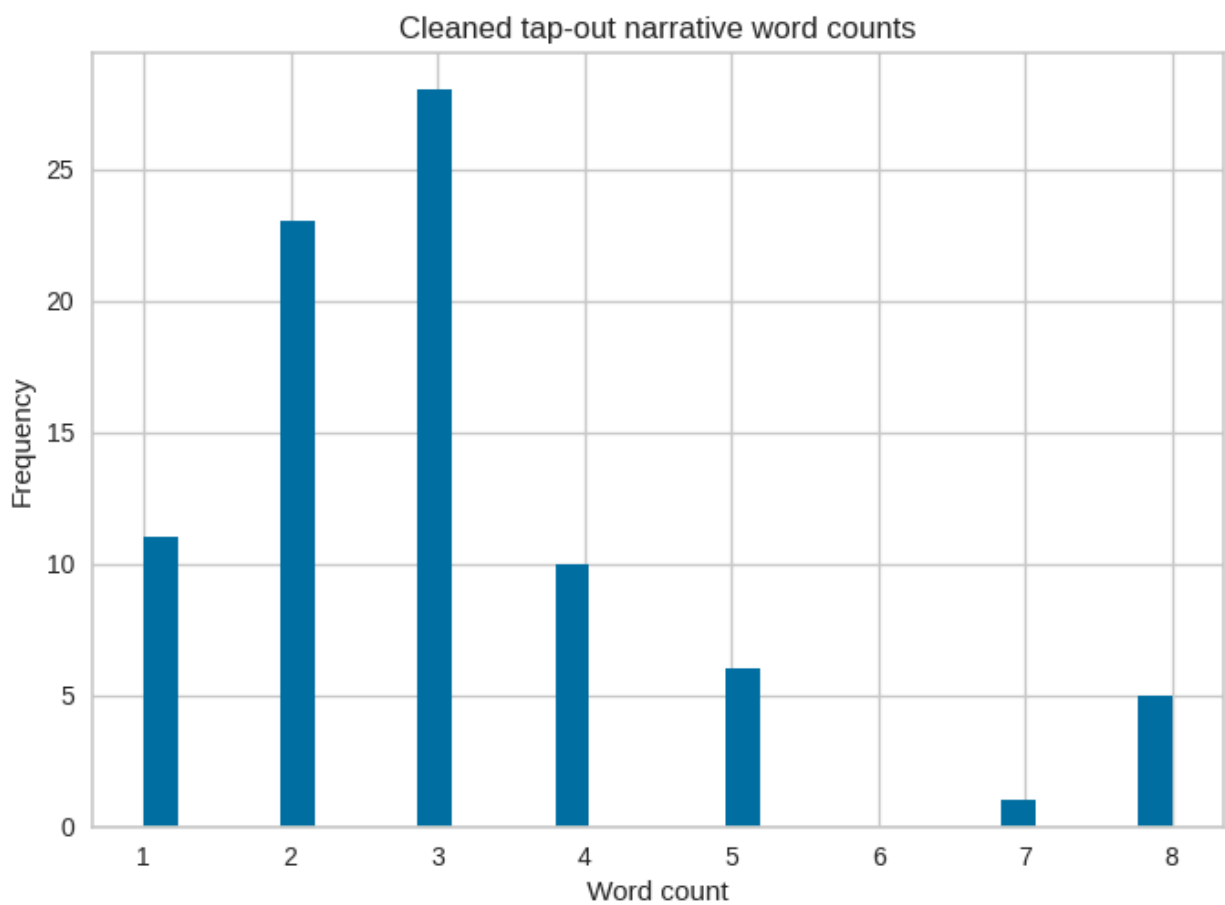
Shared GitHub Repository

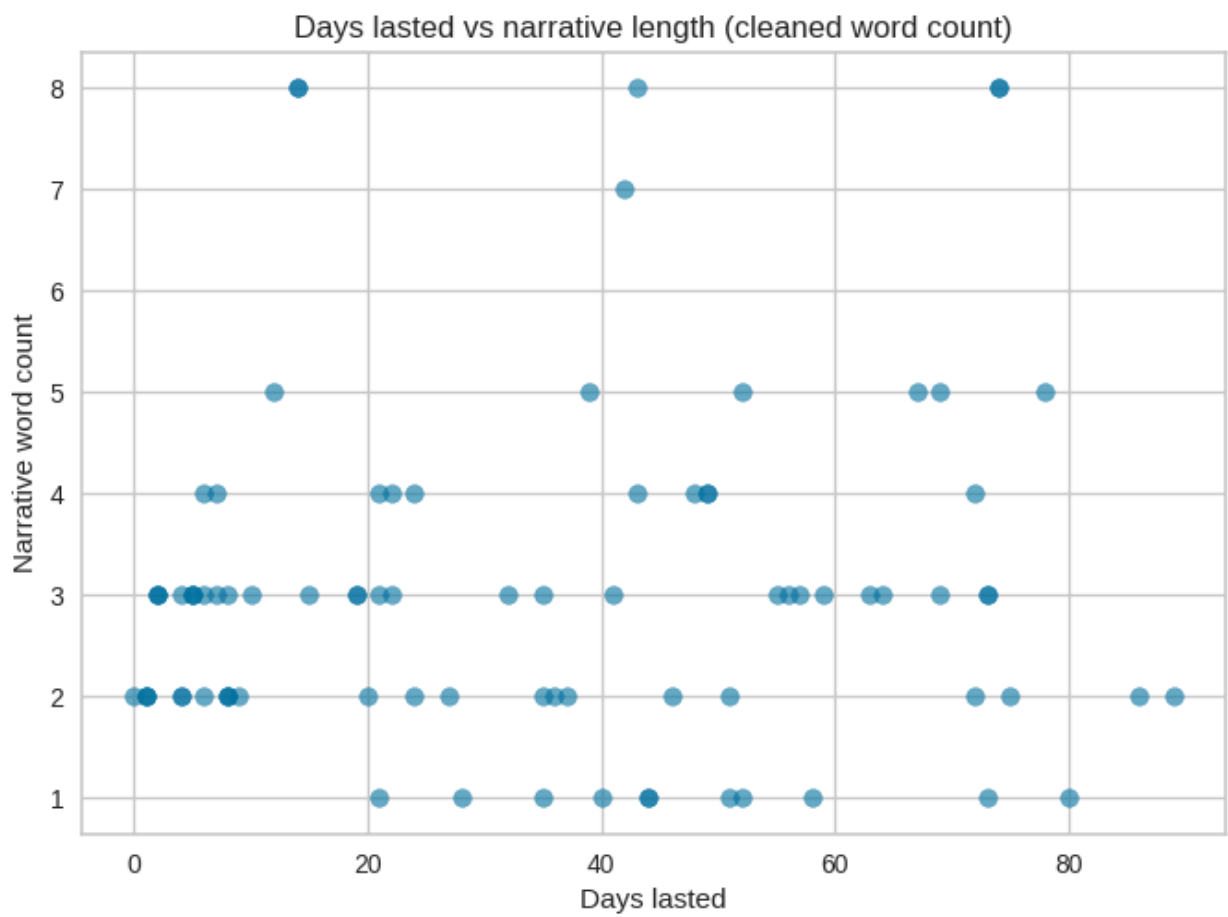
Team repo: https://github.com/MikaIsmayilov/B02_BA820_Project

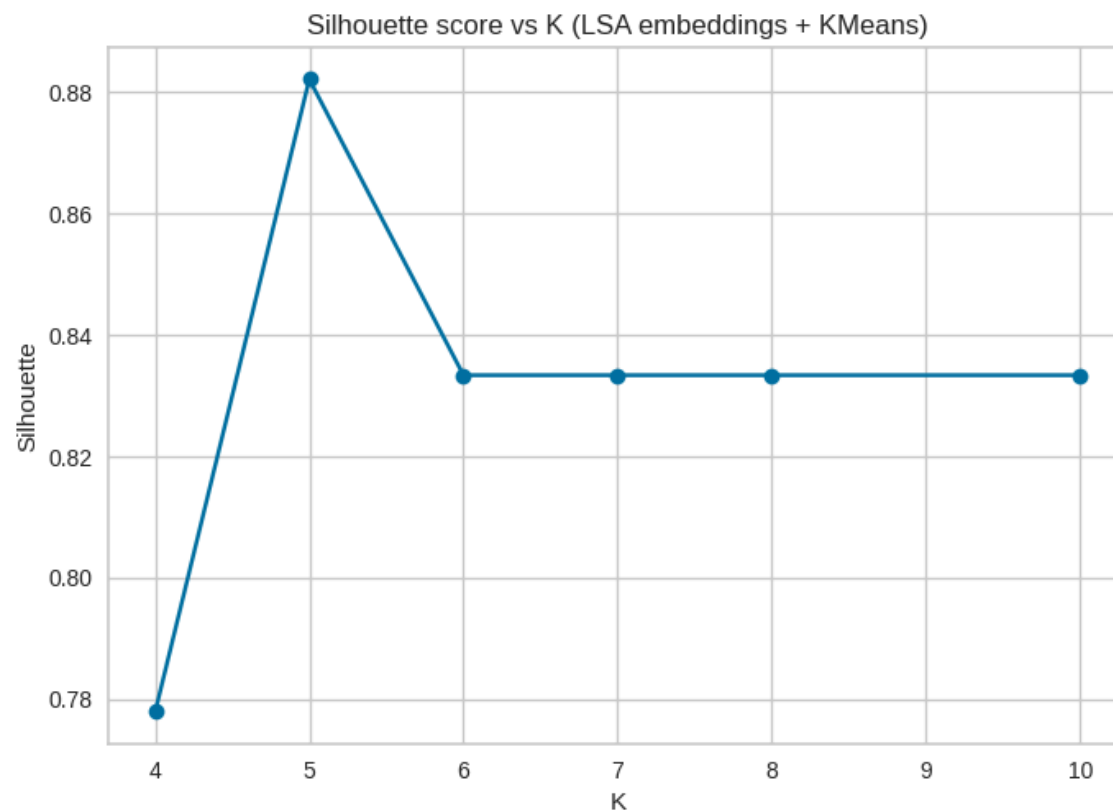
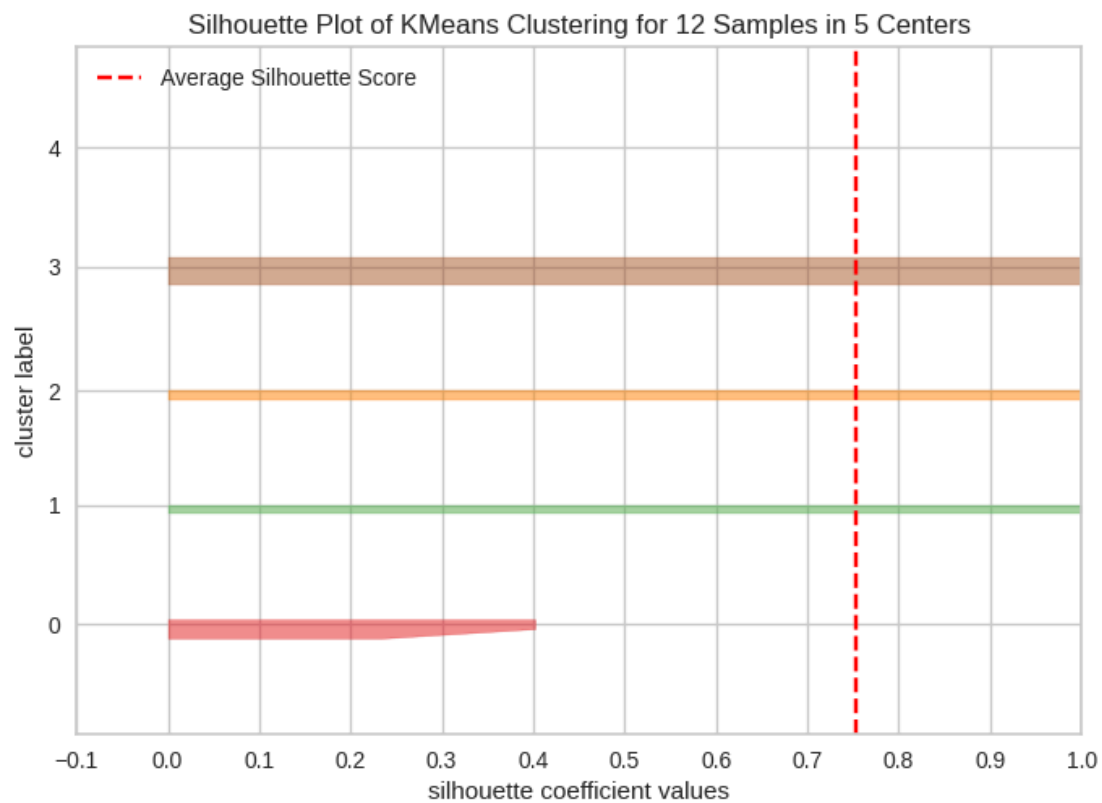
Project Proposal Team B02 BA820

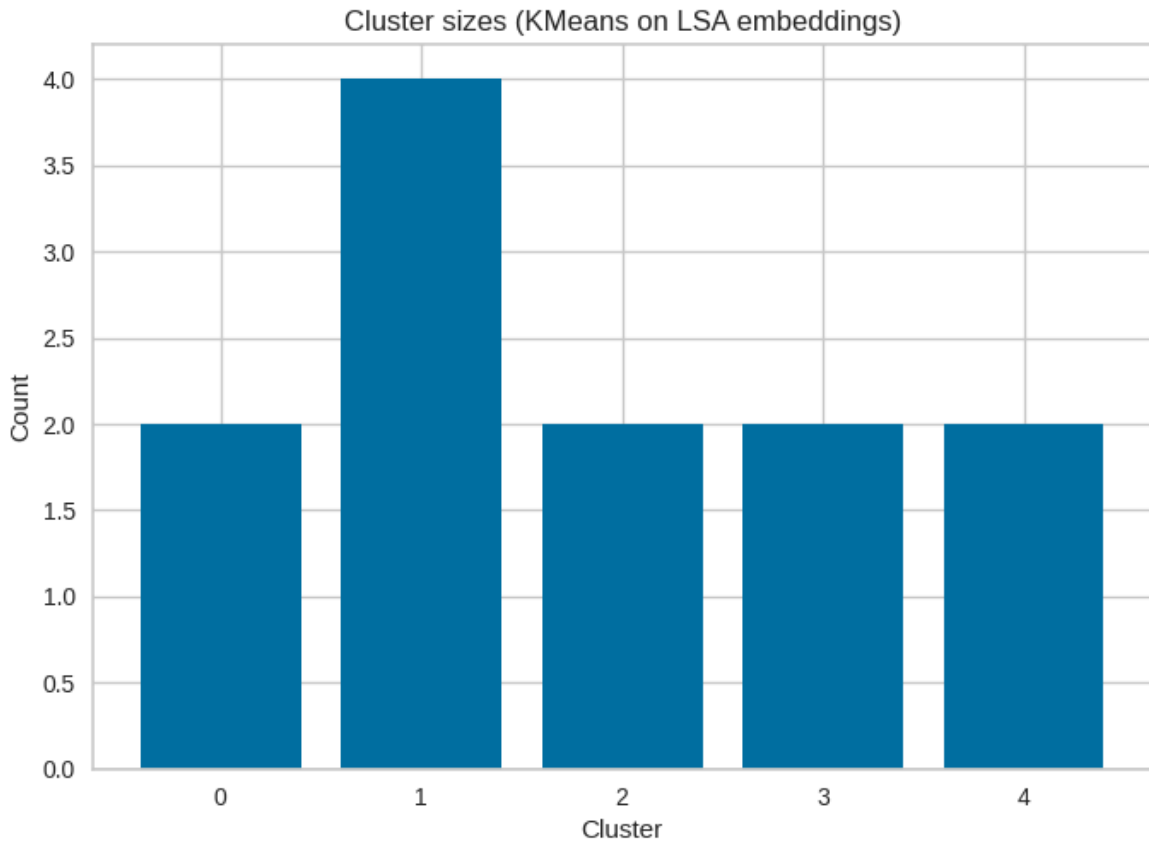
Supplemental Material:





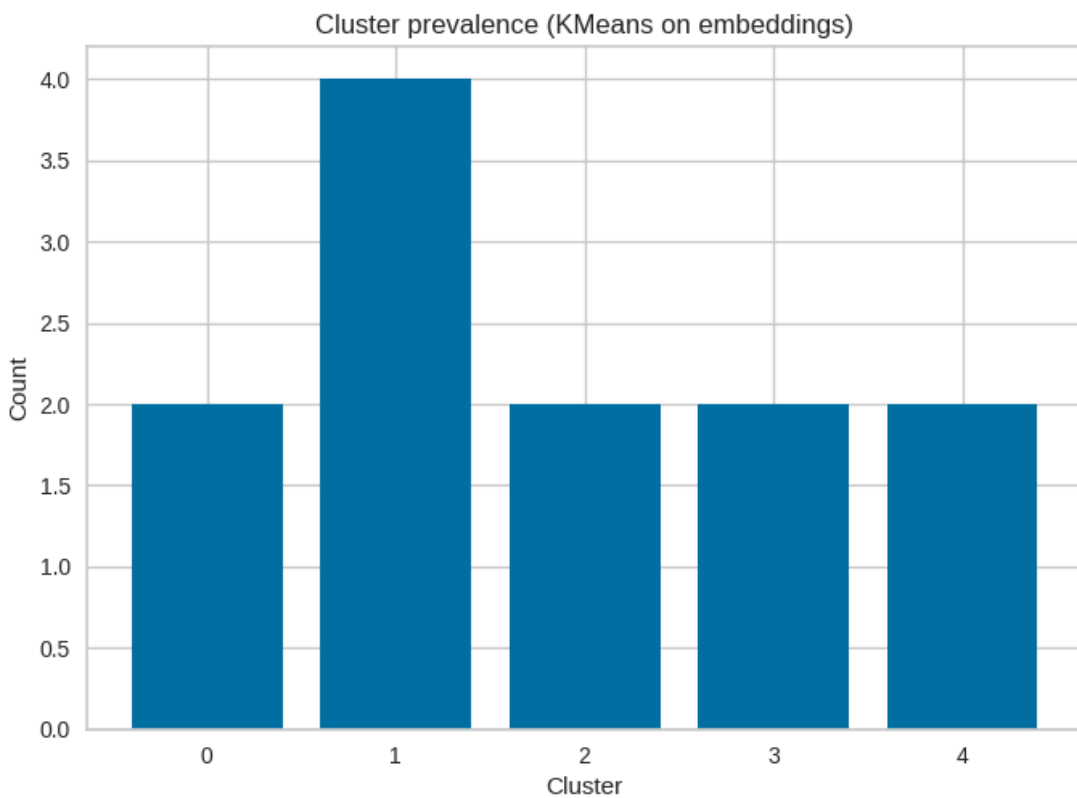
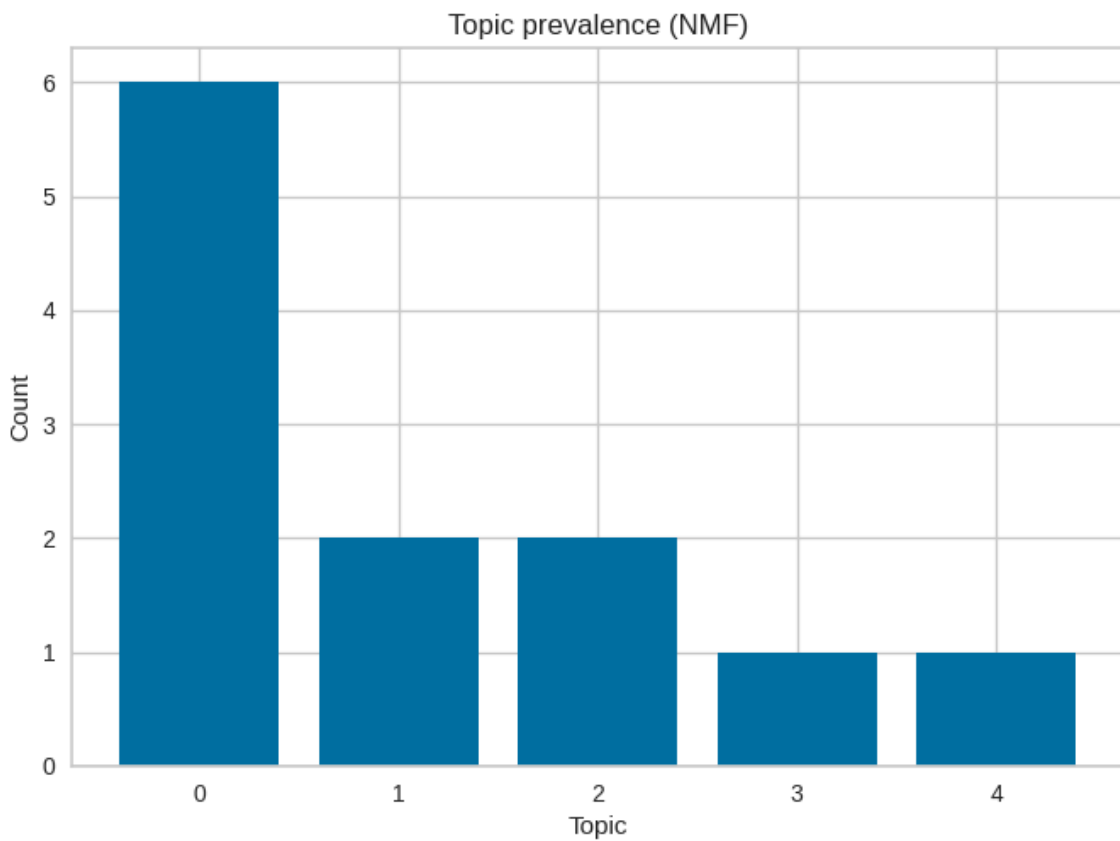


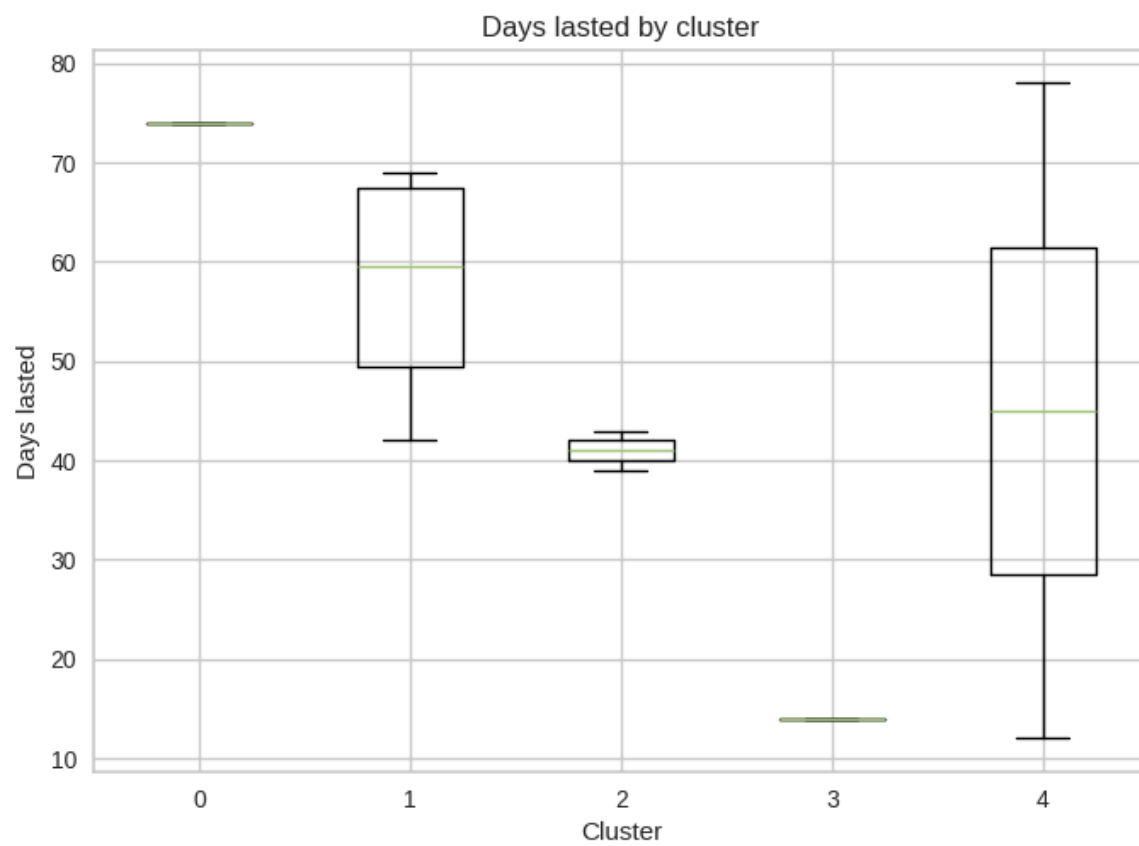
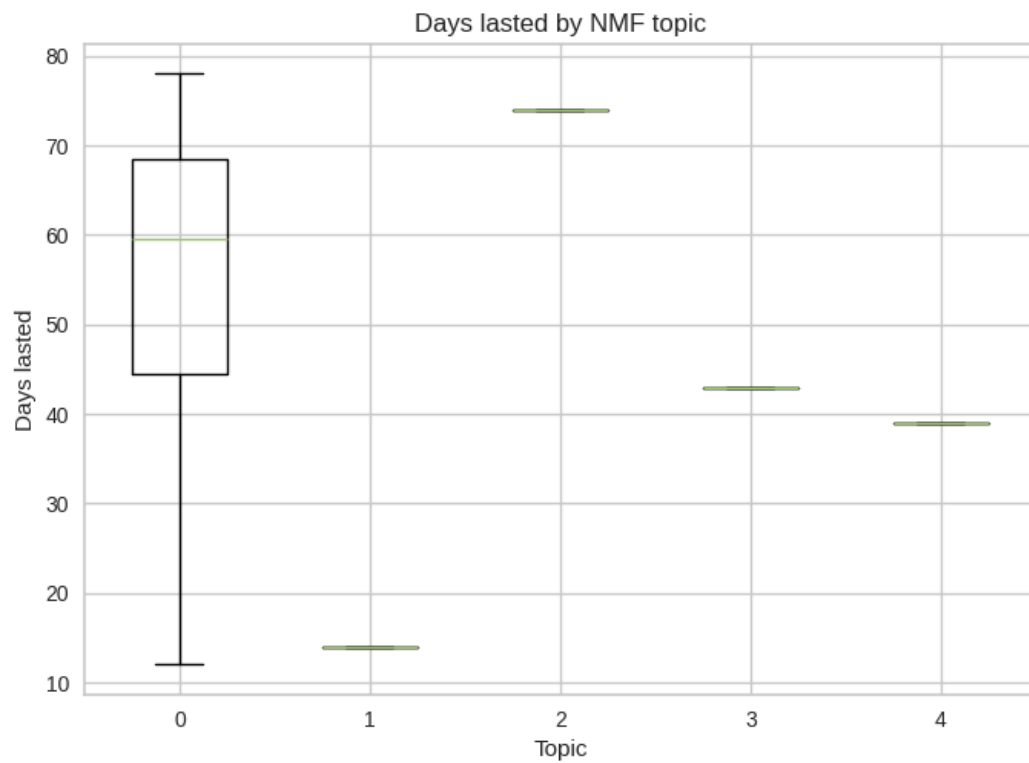




	tap_clean	topic_nmf	cluster
0	realized he actually around his mother's cancer	3	2
1	felt content what he done	4	2
2	broken teeth molars jaw pain	0	4
3	jointly decided cost game wasn't worth it	2	0
4	jointly decided cost game wasn't worth it	2	0







Process Overview (Pipeline):

1. Load survivalists table → select reason_tapped_out
2. Clean text → remove stopwords/punctuation → create bigrams
3. Filter short text (< 5 words)
4. Method 1: TF-IDF → NMF topics (+ LDA comparison)
5. Method 2: TF-IDF → SVD embeddings → KMeans → silhouette selection
6. Export theme assignments for later merging with outcomes

Use of Generative AI Tools:

I used ChatGPT to help debug my notebook by identifying errors and suggesting where specific clustering and visualization code should be placed, then I implemented the changes and reran everything. I also used it to brainstorm my write up outline and review unsupervised machine learning concepts with starter code that I adapted and validated on my own data.

Debugging: <https://chatgpt.com/share/698a7657-34e0-800c-9199-f2fcbefd71c6>

Outline brainstorming: <https://chatgpt.com/share/698a77e2-cb7c-800c-912e-7c10b2b9a367>

USML topics: <https://chatgpt.com/share/698a79c1-8948-800c-9b5f-374d1d0e1326>