Project Report

Seojin Yoo

November 2023

1. Summary

- In this investigation, we explored user preferences for planner apps through data transformation, clustering, PCA, and SVD.
- By analyzing top-rated apps, applying k-means clustering, and employing dimensionality reduction techniques, we gained insights into distinct user clusters and app preferences.

2. Introduction

This report delves into the exploration of user preferences for planner apps. Our goal is to understand patterns in user behavior, identify top-rated apps, and leverage clustering and dimensionality reduction techniques to reveal meaningful insights. The methodology involves data transformation, k-means clustering, PCA, and SVD, with a focus on interpreting the results in the context of user preferences and app ratings.

3. Methodology

3.1 Transforming Data

3.1.1 User-product matrix

Read the csv file then fill in the missing values by replacing them to ‘0’. Since there were 15 different apps in total, I decided to use all 15 apps by putting them in each column. Transformed the data into a user-product matrix.

3.1.2 Top 3 Apps and Top 3 Users

Identified the top 3 apps with the highest number of ratings. Determine the top 3 users who rated the most apps, filtering out 'A Google user,' and providing their userIds.

3.2 Clustering

3.2.1 k-means

Applied k-means clustering to the data from Q1 and clustered the users. Measured the inertia for each value of k. Plotted the resulting inertia scores for each choice of k.

3.2.2 Finding appropriate value of k

Looked for an "elbow" in the inertia plot, where the rate of decrease of inertia slows down. Found a point where adding more clusters doesn't significantly reduce inertia. Used 16.

3.2.3 Cluster with chosen value of k

Created a mapping between userName and cluster labels. Mapped the cluster labels back to the original DataFrame. Found mean ratings and the top three highest rated apps for each cluster.

3.3 Principle Component Analysis

3.3.1 Transpose matrix

Transpose the matrix from Q1. Mean center the data.

3.3.2 Apply PCA (k = 2)

Apply PCA with the number of components k = 2. Create a dataframe with the principal components and display the results.

3.3.3 Data Visualization

Plot the results with different colors for each appId. There are 15 different colors for each app.

3.3.4 "Intrinsic" Dimensionality

Determine the "intrinsic" dimensionality. Calculate explained variance ratio. Find the number of principle components that are needed to explain 80% and 40% of the variance. Compare with k = 2.

3.4 Singular Value Decomposition

3.4.1 Apply SVD (k = 15)

Apply SVD with number of components k = 15. In the question, it says k = 128; however, for this dataset since there are 15 apps, we are using the maximum k value, 15. Plot the resulting singular values.

3.4.2 Sum of explained_variance_ratio

Due to the same reason above, instead of using k = [2, 4, 8, 16, 32, 64, 128], for each of the values of k = [2, 4, 8, 15], calculate and print the sum of explained variance ratio for each k.

3.4.3 Transforming Data (k = 2)

Apply SVD with k = 2 and transform the data.

3.4.4 Analysis

Plot the results (for k = 2) and color the users by the cluster memberships. Discuss patterns based on the 15 different apps and compare them to the previous analysis.

4. Result

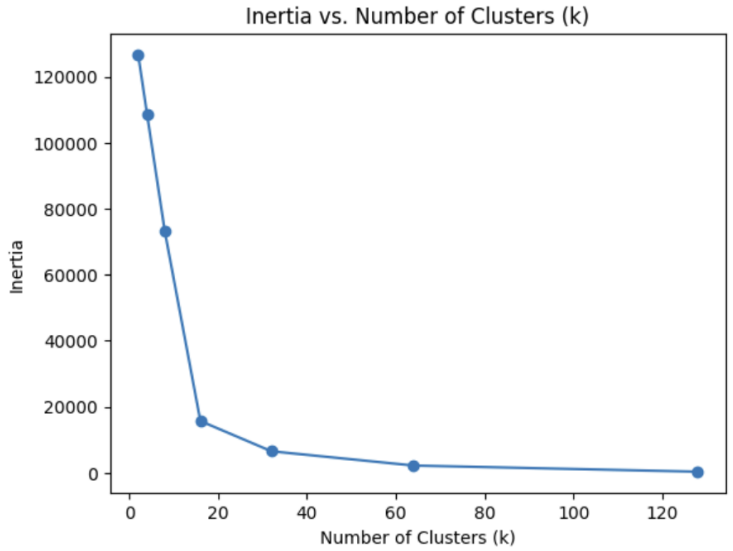
1 Transforming Data

(a) Top 3 apps in terms of number of users that have rated them are forest, Microsoft, and Any.do. Forest was rated 947 times. Microsoft was rated 943 times. Any.do was rated 939 times.

(b) The userIds for the top 3 users that have rated the greatest number of apps are Brajeswar Das, Chris, and Kelly Madison. All three of them rated 4 apps.

2 Clustering

(a) The resulting inertia scores for each choice of k is plotted on the below graph.



(b) To choose an appropriate k is to look for an "elbow" in the inertia plot, where the rate of decrease of inertia slows down. The idea is to find a point where adding more clusters doesn't significantly reduce inertia. k = 16 seems reasonable for this case.

(c) For each of the resulting clusters, the top three apps that are highest rated by the users in the cluster are shown below. The k-means clustering results reveal distinct clusters of users with similar preferences for planner apps. Analyzing the top-rated apps in each cluster suggests that these clusters may correspond to specific user preferences or usage patterns such as wanting minimalism or apps that apparently have less ads for premium versions.

	Cluster ('score', 0)	Cluster ('score', 1)	Cluster ('score', 2)	Cluster ('score', 3)	Cluster ('score', 4)	Cluster ('score', 5)
0	(com.gmail.jmartindev.timetune, 1.490514905149...	(prox.lab.calclock, 3.6983408748114632)	(cc.forestapp, 3.541160593792173)	(com.anydo, 5.0)	(com.tasks.android, 3.7739403453689166)	(com.todoist, 3.5744089012517386)
1	(com.levor.liferpgtasks, 1.4299065420560748)	(com.habitnow, 3.5)	(com.tasks.android, 3.0)	(com.microsoft.todos, 3.5453315290933696)	(com.anydo, 3.0)	(com.gmail.jmartindev.timetune, 3.0)
2	(com.habitnow, 1.3442622950819672)	(com.microsoft.todos, 3.0)	(prox.lab.calclock, 3.0)	(cc.forestapp, 3.5)	(com.artfulagenda.app, 3.0)	(com.microsoft.todos, 3.0)

Cluster ('score', 6)	Cluster ('score', 7)	Cluster ('score', 8)	Cluster ('score', 9)	Cluster ('score', 10)	Cluster ('score', 11)
(com.habitrpg.android.habitica, 3.549315068493...	(com.appgenix.bizcal, 3.5454545454545454)	(com.anydo, 3.5297297297297296)	(com.anydo, 4.0)	(com.oristats.habitbull, 3.5658807212205272)	(com.tasks.android, 4.666666666666667)
(com.ticktick.task, 3.3333333333333335)	(com.ticktick.task, 3.0)	(com.levor.liferpgtasks, 3.1)	(prox.lab.calclock, 4.0)	(com.appgenix.bizcal, 3.5)	(com.levor.liferpgtasks, 4.2)
(com.habitnow, 3.0)	(com.tasks.android, 2.0)	(com.artfulagenda.app, 3.0)	(com.ticktick.task, 3.5816901408450703)	(com.habitnow, 2.6666666666666665)	(cc.forestapp, 4.0)

Cluster ('score', 12)	Cluster ('score', 13)	Cluster ('score', 14)	Cluster ('score', 15)
(com.gmail.jmartindev.timetune, 4.097966728280...	(com.ticktick.task, 4.5)	(com.artfulagenda.app, 4.483606557377049)	(com.appxy.planner, 3.5665722379603397)
(prox.lab.calclock, 3.3333333333333335)	(com.habitnow, 4.369077306733167)	(com.levor.liferpgtasks, 4.0)	(cc.forestapp, 2.0)
(com.habitnow, 3.0)	(com.anydo, 4.0)	(cc.forestapp, nan)	(com.gmail.jmartindev.timetune, 2.0)

Below is the matrix for mean_ratings_by_cluster.

	score											
cluster	0	1	2	3	4	5	6	7	8	9	10	11
appId												
cc.forestapp	1.000000	2.000000	3.541161	3.500000	2.000000	NaN	NaN	NaN	1.750000	NaN	NaN	4.000000
com.anydo	1.000000	NaN	NaN	5.000000	3.000000	1.666667	NaN	NaN	3.529730	4.000000	NaN	2.500000
com.appgenix.bizcal	1.000000	NaN	1.000000	NaN	NaN	NaN	1.000000	3.545455	2.464286	1.000000	3.500000	4.000000
com.appxy.planner	1.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2.500000	NaN	NaN	1.000000
com.artfulagenda.app	1.318182	NaN	NaN	NaN	3.000000	NaN	NaN	NaN	3.000000	NaN	NaN	NaN
com.gmail.jmartindev.timetune	1.490515	2.000000	NaN	2.000000	NaN	3.000000	2.000000	NaN	1.966667	2.000000	2.000000	NaN
com.habitnow	1.344262	3.500000	NaN	NaN	NaN	NaN	3.000000	NaN	NaN	3.000000	2.666667	3.000000
com.habitrpg.android.habitica	1.000000	1.500000	NaN	1.000000	1.000000	1.000000	3.549315	NaN	2.105263	1.000000	2.000000	2.200000
com.levor.liferpgtasks	1.429907	2.000000	NaN	NaN	NaN	NaN	2.000000	NaN	3.100000	1.000000	NaN	4.200000
com.microsoft.todos	1.005051	3.000000	NaN	3.545332	2.000000	3.000000	2.000000	NaN	2.142857	3.000000	NaN	NaN
com.oristats.habitbull	1.000000	1.500000	NaN	3.000000	NaN	3.000000	1.000000	NaN	2.250000	3.500000	3.565881	2.666667
com.tasks.android	1.000000	NaN	3.000000	2.000000	3.773940	2.600000	NaN	2.000000	1.933333	1.750000	NaN	4.666667
com.ticktick.task	1.000000	3.000000	NaN	2.000000	1.666667	2.000000	3.333333	3.000000	2.192308	3.581690	1.000000	2.000000
com.todoist	1.000000	1.333333	NaN	2.166667	2.333333	3.574409	2.000000	NaN	2.350000	1.666667	NaN	2.666667
prox.lab.calclock	1.000000	3.698341	3.000000	2.333333	1.000000	2.000000	3.000000	2.000000	1.840000	4.000000	NaN	3.600000

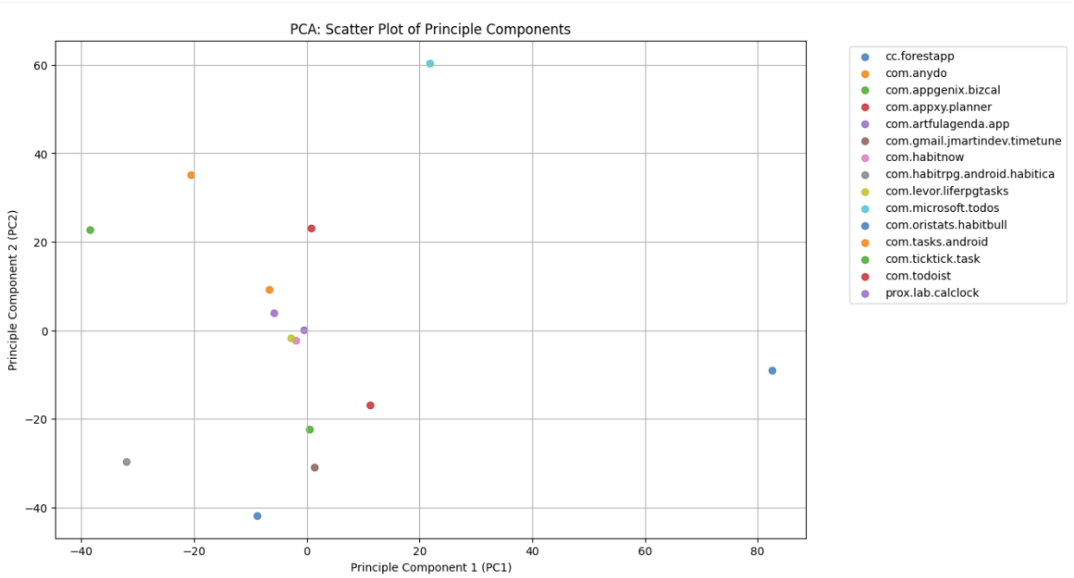
12	13	14	15
2.500000	NaN	NaN	2.000000
2.000000	4.000000	NaN	NaN
2.500000	NaN	NaN	NaN
NaN	2.333333	NaN	3.566572
NaN	NaN	4.483607	1.000000
4.097967	3.500000	NaN	2.000000
3.000000	4.369077	NaN	NaN
3.000000	3.000000	NaN	1.000000
1.000000	4.000000	4.000000	NaN
NaN	NaN	NaN	NaN
3.000000	3.500000	NaN	NaN
3.000000	2.500000	NaN	NaN
NaN	4.500000	NaN	1.000000
1.333333	NaN	NaN	NaN
3.333333	NaN	NaN	NaN

3 Principle Component Analysis

(a) Transposed the matrix from Q1 and mean centered the data.

(b) Applied PCA with number of components k = 2.

(c) Below is the plot of the results. I colored each 15 apps individually because there aren't too many apps and there aren't any variables to group them by. Examining the graph, I can see that most of the principle component 1s are close to 0. PC2 seems to be distributed uniformly.



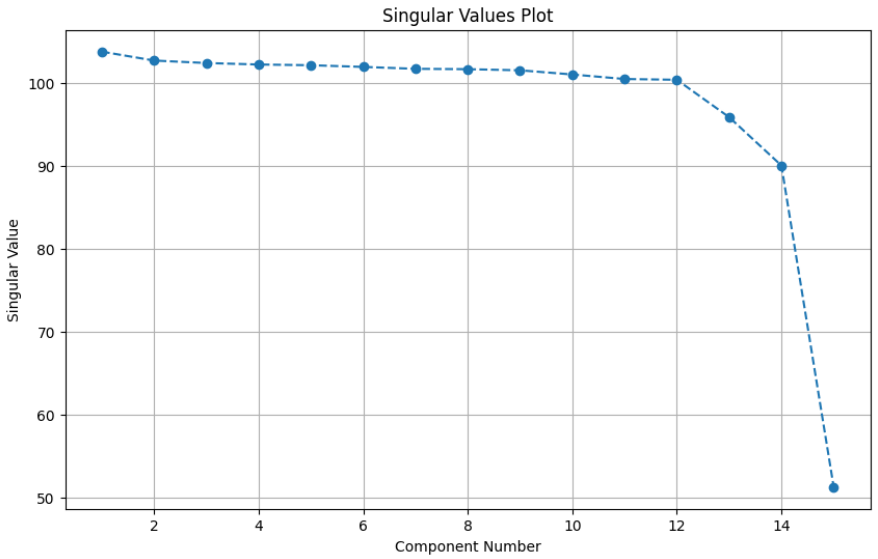
(d) Number of principal components to explain 80% of the variance: 11

Number of principal components to explain 40% of the variance: 6

This implies that the number of components, k=2, is not enough to explain a large portion of the variance. It suggests that the original PCA visualization may not be the most informative, and a lower-dimensional representation with more components could be used.

4 Singular Value Decomposition

(a) Below is the plot of the resulting singular_values. I applied SVD with the number of components k = 15 since there are only 15 different apps and therefore 15 components.



(b) For each of the values of k = [2, 4, 8, 15], the sum of the explained_variance_ratio is as below. In Q2 we found that with the values of k = [2, 4, 8, 16] the inertia drops quickly and starts an elbow, showing that 16 is about the limit where adding more clusters does not provide as large benefits. The results here are similar, with the sums of the explained variance in the same range are enough to account for the variance and accurately model the data.

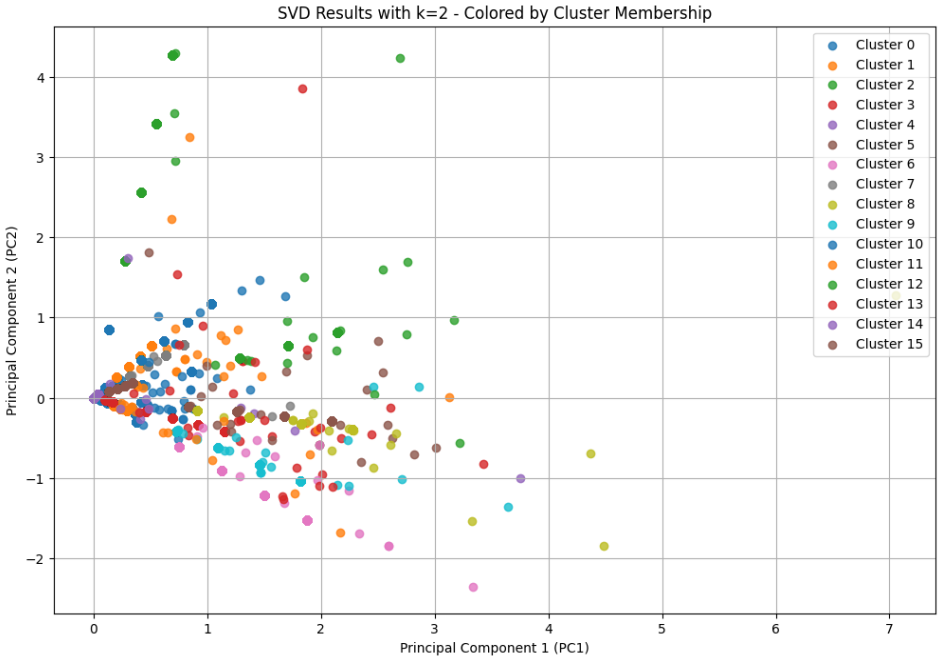
```
Sum of Explained Variance Ratio for k=2: 0.1051145759673921
Sum of Explained Variance Ratio for k=4: 0.25745147932262386
Sum of Explained Variance Ratio for k=8: 0.5623148123107383
Sum of Explained Variance Ratio for k=15: 0.9999999999999951
```

(c) Applied SVD with k = 2 and transformed the data.

```
[[ 1.03459062  1.17529792]
 [ 1.28464818  0.48880313]
 [ 0.91177031 -0.16118716]
 ...
 [ 0.13789614  0.85370548]
 [ 1.50325621 -1.21589593]
 [ 0.23205293 -0.13136337]]
```

(d) Below is the plot of the results (for k = 2). The clusters look more distinct and spread out in different directions compared to the previous analysis. It seems

like the components are concentrated to 0 for both PC2 and PC1. Cluster 0 seems to move upwards and to the right while cluster 14 seems to move in the downward right direction.



5. Conclusion

Through rigorous data analysis, we discovered notable patterns in user preferences for planner apps. K-means clustering revealed distinct user clusters with similar preferences, and PCA provided insights into the inherent structure of the data. SVD further supported these findings, emphasizing the importance of a lower-dimensional representation. The top-rated apps and users identified contribute to a comprehensive understanding of the planner app landscape. As we conclude, these insights pave the way for informed decisions in app development and user engagement strategies. In summary, this report successfully navigated through various data analysis techniques to uncover valuable insights into user behavior and preferences for planner apps. The combination of clustering and dimensionality reduction

techniques provided a holistic view, laying the foundation for informed decision-making in the realm of app development and user engagement.

6. References

Below are the links for the code references I used for this project.

1. https://huggingface.co/docs/datasets/v1.5.0/processing.html#exporting-a-dataset-to-csv-or-to-python-objects
2. https://huggingface.co/docs/datasets/v1.5.0/package_reference/main_classes.html#datasets.Dataset.to_csv
3. https://huggingface.co/docs/datasets/v1.5.0/package_reference/main_classes.html#datasets.Dataset.to_pandas
4. https://stackoverflow.com/questions/51342408/how-do-i-install-python-packages-in-googles-colab