# DATA301 Project Report

Robert Ivill 46012819

## Abstract:

This project uses the Twitch dataset from the Recommender Systems and Personalization Datasets, which contains data about user interactions with streamers on the live streaming platform Twitch. The research question is: Can we categorize streamers based on their viewer engagement and streaming activities? To answer this question, we use the K-means clustering algorithm to analyse the various features that were derived from the dataset, such as the mean watch time, total interactions, total streams, and number of unique users per streamer. By clustering streamers based on these metrics, we identified patterns and were able to categorize streamers. These results provide insight into how streaming behaviors can affect how users interact and engage with these streamers. This analysis would allow streamers to see how their activity on Twitch affects their performance with viewers, and potentially allow them to schedule and use strategies that would improve this performance.

## Introduction:

In this project, we are using the Twitch Dataset from the Recommender Systems and Personalization Datasets provided by the University of California San Diego (UCSD) research lab. This dataset contains information about users' interactions with streamers, specifically streamers with 100,000 or more followers on Twitch. Over 43 days, they recorded users that were active in a streamers chat, updating every 10 minutes whether they were still active or not. The data contains an anonymous User ID, a stream ID, the streamer's username, the time that the interaction began, and the time that the interaction finished. The time start and stop was recorded as an integer that represents which 10-minute period the interaction was recorded on for the study.

The algorithm that we will be applying is the K-means clustering algorithm to explore categories between streamers and their features. This algorithm achieves this by minimizing the distance between datapoints and a set number of centroids to find clusters of data that can be categorized together. This way, we can see the relationships between different streamers and how their viewers interact with them. To do this, however, we will have to group the data by each streamer and calculate new features that will be useful for this categorization. We also must select a number of clusters to use, which is done by using the elbow method for KMeans, which minimizes distance within clusters whilst minimizing the number of clusters to prevent overfitting.

The research question of this project is relevant to me, as I have a fond interest in viewing and interacting with various streamers on Twitch. I know that not every streamer interacts with their viewers in the same way, and vice versa. Some streamers can stream everyday and get a healthy number of viewers and interactions every time, whereas others may choose to stream on a rare occasion, resulting in a larger number of viewers each time they stream. It will be interesting to see if the data will be able to quantify the differences between these

types of streamers, seeing if there is an objectively best way to stream to increase interactions with users.

The research question of this project is “Can we categorize streamers based on their viewer engagement and streaming activities?”. This is relevant to the dataset as it shows the period of a viewer’s engagement with each streamer, where a streamer would ideally want this value to be as high as possible. By finding various features, such as the number of unique users, total interactions, and total streams per streamer, we will be able to see how these streamers can be categorized. The K-means clustering algorithm will take these features and show us clusters of streamers together, allowing us to categorize each streamer type by their cluster.

## Experimental Design and Methods:

To find a robust answer to our research question, I had to create a program design that allowed us to mend and visualize the data into a useful state. This began with gathering the data into a Dask Dataframe with a blocksize of 1MB, making sure that we are using a sensible amount of parallelism for our dataframe. The next step was to clean the data, which involved adding column names to the data, dropping missing values from the dataframe, and adding a ‘WatchTime’ feature to the dataframe. This was calculated by subtracting the start time from the finish time and could easily be converted into seconds by multiplying by 600 (Since the base value was 10 minutes for start and finish times). To answer the research question, we had to organize the data to give us values for each streamer. The Dask function ‘groupby’ allowed for organizing data per streamer, then we could compute the mean watch time, total interactions, total streams, and the number of unique users per streamer.

As the data was now organized and ready, we could apply K-means clustering and visualization of our results. As the data had a large variance, I decided to do the clustering for both the original data and a scaled version of the data, which was scaled using the StandardScaler module. From here, KMeans was created for 4 clusters, which was calculated using the elbow method for KMeans, and used to fit on the streamer features dataframes. The cluster value was added to the dataframes for each streamer, then we moved on to data visualization and analysis. Matplotlib and seaborn were used to create a pair plot that showed each feature against each other and simultaneously show the clusters in the data. The describe function meant that we could see the various statistics for each cluster group among the scaled and unscaled data, allowing us to move on to the analysis of results.

A variety of code and libraries were used to implement methods within this project. This is a list of them and how they helped:

URLLib.Request – Used to retrieve the data from its URL and load it into a file.

Dask.dataframe – Used for most of the large data. The file was loaded into a dataframe to be tidied and modified.

Pandas – Used for the streamer features dataframe, as the dataset was not large enough to warrant the use of Dask. This dataframe was used for clustering algorithm and visualization.

SciKitLearn MinMaxScaler – This was used for scaling the dataset to a distribution between 0 and 1 and allowed for a different analysis of the data when it came to clustering.

SciKitLearn KMeans – This was used to perform cluster analysis on the data. The data was plotted to a graph comparing the number of clusters to its clustering score or the sum of the squared distances between clusters. A value of 4 was chosen. The fit_predict function was used to find the cluster values and then add them to the data.

Seaborn and Matplotlib – These were used for the visualization of the data into graphs. Seaborn was used to create pair plots of the data.

## Results:

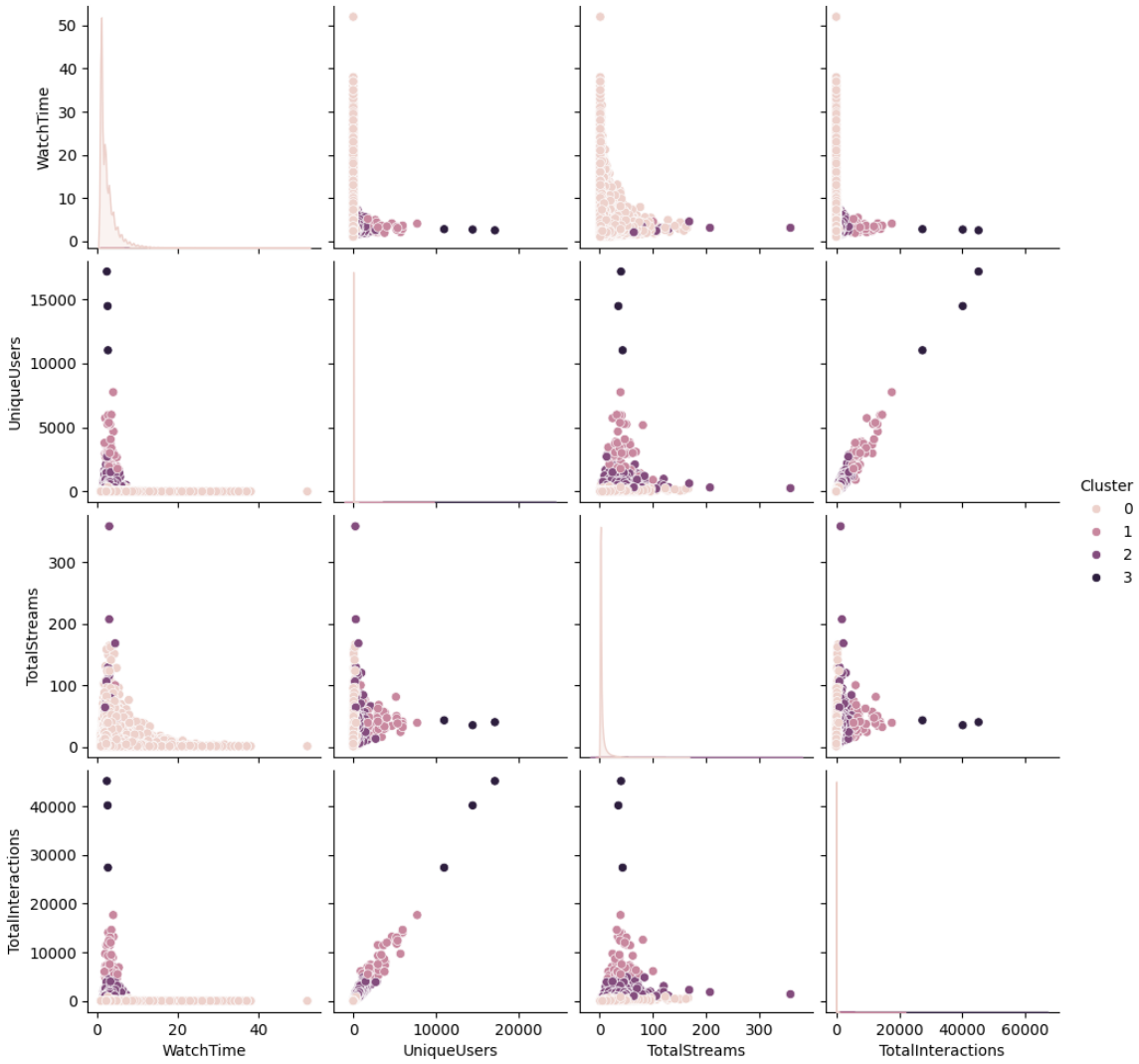


Figure 1: Pair Plot of streamer features

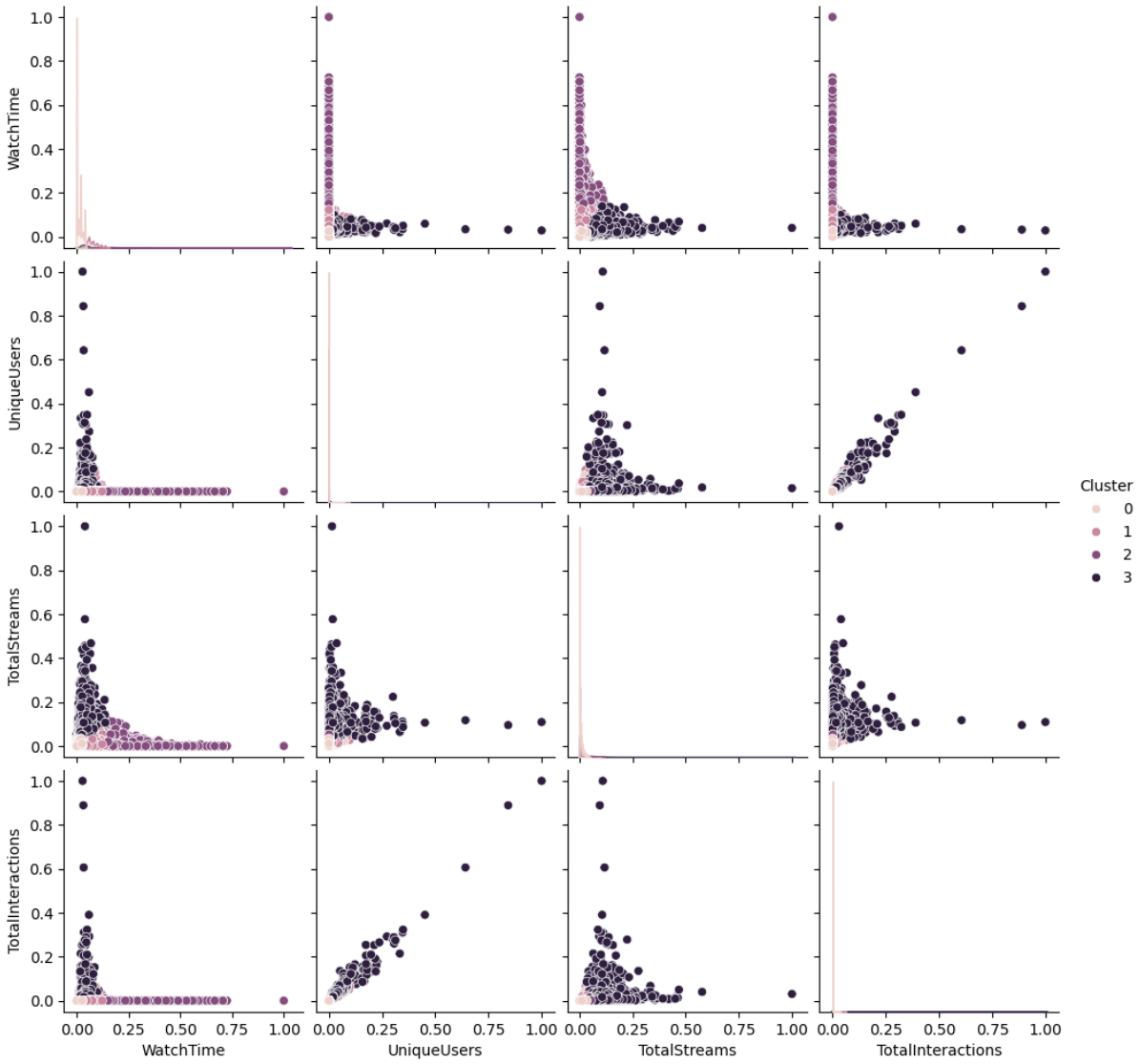


Figure 2: Pair plot of streamer features, scaled from 0 to 1

Cluster 0:

| | WatchTime | UniqueUsers | TotalStreams | TotalInteractions | Cluster |
|---|---|---|---|---|---|
| count | 162118.000000 | 162118.000000 | 162118.000000 | 162118.000000 | 162118.0 |
| mean | 2.694982 | 5.978448 | 4.431587 | 10.725589 | 0.0 |
| std | 2.511540 | 22.502799 | 7.410929 | 45.302153 | 0.0 |
| min | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 0.0 |
| 25% | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 0.0 |
| 50% | 2.000000 | 1.000000 | 2.000000 | 2.000000 | 0.0 |
| 75% | 3.214286 | 3.000000 | 4.000000 | 5.000000 | 0.0 |
| max | 52.000000 | 583.000000 | 166.000000 | 910.000000 | 0.0 |

Cluster 1:

| | WatchTime | UniqueUsers | TotalStreams | TotalInteractions | Cluster |
|---|---|---|---|---|---|
| count | 53.000000 | 53.000000 | 53.000000 | 53.000000 | 53.0 |
| mean | 3.315737 | 3205.981132 | 44.056604 | 7968.358491 | 1.0 |
| std | 0.883841 | 1476.824146 | 16.750907 | 3166.900311 | 0.0 |
| min | 1.885243 | 903.000000 | 15.000000 | 4610.000000 | 1.0 |
| 25% | 2.619238 | 1989.000000 | 33.000000 | 5599.000000 | 1.0 |
| 50% | 3.211518 | 2981.000000 | 41.000000 | 6623.000000 | 1.0 |
| 75% | 3.935585 | 3787.000000 | 51.000000 | 9476.000000 | 1.0 |
| max | 5.429735 | 7747.000000 | 100.000000 | 17633.000000 | 1.0 |

Cluster 2:

| | WatchTime | UniqueUsers | TotalStreams | TotalInteractions | Cluster |
|---|---|---|---|---|---|
| count | 451.000000 | 451.000000 | 451.000000 | 451.000000 | 451.0 |
| mean | 3.234251 | 717.077605 | 42.345898 | 1724.957871 | 2.0 |
| std | 0.817084 | 387.526757 | 24.812812 | 893.203886 | 0.0 |
| min | 1.661965 | 185.000000 | 4.000000 | 764.000000 | 2.0 |
| 25% | 2.685509 | 456.000000 | 31.000000 | 1057.000000 | 2.0 |
| 50% | 3.168806 | 598.000000 | 40.000000 | 1443.000000 | 2.0 |
| 75% | 3.695084 | 859.500000 | 49.000000 | 2052.500000 | 2.0 |
| max | 7.203579 | 2842.000000 | 358.000000 | 4990.000000 | 2.0 |

Cluster 3:

| | WatchTime | UniqueUsers | TotalStreams | TotalInteractions | Cluster |
|---|---|---|---|---|---|
| count | 3.000000 | 3.000000 | 3.000000 | 3.000000 | 3.0 |
| mean | 2.622040 | 14208.333333 | 39.333333 | 37547.333333 | 3.0 |
| std | 0.137951 | 3079.223008 | 4.041452 | 9169.284450 | 0.0 |
| min | 2.469741 | 11011.000000 | 35.000000 | 27362.000000 | 3.0 |
| 25% | 2.563752 | 12735.500000 | 37.500000 | 33749.000000 | 3.0 |
| 50% | 2.657764 | 14460.000000 | 40.000000 | 40136.000000 | 3.0 |
| 75% | 2.698190 | 15807.000000 | 41.500000 | 42640.000000 | 3.0 |
| max | 2.738616 | 17154.000000 | 43.000000 | 45144.000000 | 3.0 |

Figure 3: Statistics for different clusters against the streamers features

Cluster 0:

| | WatchTime | UniqueUsers | TotalStreams | TotalInteractions | Cluster |
|---|---|---|---|---|---|
| count | 116172.000000 | 116172.000000 | 116172.000000 | 116172.000000 | 116172.0 |
| mean | 0.011436 | 0.000104 | 0.004561 | 0.000061 | 0.0 |
| std | 0.013623 | 0.000608 | 0.008038 | 0.000307 | 0.0 |
| min | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.0 |
| 25% | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.0 |
| 50% | 0.004902 | 0.000000 | 0.000000 | 0.000000 | 0.0 |
| 75% | 0.019608 | 0.000058 | 0.005602 | 0.000044 | 0.0 |
| max | 0.045752 | 0.071999 | 0.050420 | 0.038965 | 0.0 |

Cluster 1:

| | WatchTime | UniqueUsers | TotalStreams | TotalInteractions | Cluster |
|---|---|---|---|---|---|
| count | 30765.000000 | 30765.000000 | 30765.000000 | 30765.000000 | 30765.0 |
| mean | 0.079823 | 0.000173 | 0.008690 | 0.000130 | 1.0 |
| std | 0.026553 | 0.001272 | 0.011637 | 0.000853 | 0.0 |
| min | 0.044118 | 0.000000 | 0.000000 | 0.000000 | 1.0 |
| 25% | 0.058824 | 0.000000 | 0.000000 | 0.000000 | 1.0 |
| 50% | 0.075415 | 0.000000 | 0.002801 | 0.000044 | 1.0 |
| 75% | 0.098039 | 0.000117 | 0.011204 | 0.000111 | 1.0 |
| max | 0.150895 | 0.097884 | 0.095238 | 0.065481 | 1.0 |

Cluster 2:

| | WatchTime | UniqueUsers | TotalStreams | TotalInteractions | Cluster |
|---|---|---|---|---|---|
| count | 5617.000000 | 5617.000000 | 5617.000000 | 5617.000000 | 5617.0 |
| mean | 0.218682 | 0.000037 | 0.004550 | 0.000046 | 2.0 |
| std | 0.074123 | 0.000160 | 0.009566 | 0.000132 | 0.0 |
| min | 0.150327 | 0.000000 | 0.000000 | 0.000000 | 2.0 |
| 25% | 0.166667 | 0.000000 | 0.000000 | 0.000000 | 2.0 |
| 50% | 0.196078 | 0.000000 | 0.000000 | 0.000000 | 2.0 |
| 75% | 0.241830 | 0.000000 | 0.005602 | 0.000044 | 2.0 |
| max | 1.000000 | 0.005597 | 0.112045 | 0.002747 | 2.0 |

Cluster 3:

| | WatchTime | UniqueUsers | TotalStreams | TotalInteractions | Cluster |
|---|---|---|---|---|---|
| count | 10071.000000 | 10071.000000 | 10071.000000 | 10071.000000 | 10071.0 |
| mean | 0.039477 | 0.006017 | 0.078888 | 0.005224 | 3.0 |
| std | 0.019174 | 0.022739 | 0.037641 | 0.021742 | 0.0 |
| min | 0.000000 | 0.000000 | 0.030812 | 0.000288 | 3.0 |
| 25% | 0.025772 | 0.000933 | 0.053221 | 0.000731 | 3.0 |
| 50% | 0.036909 | 0.001749 | 0.070028 | 0.001329 | 3.0 |
| 75% | 0.049753 | 0.004198 | 0.095238 | 0.003278 | 3.0 |
| max | 0.151961 | 1.000000 | 1.000000 | 1.000000 | 3.0 |

Figure 4: Statistics for different clusters against the scaled streamers features

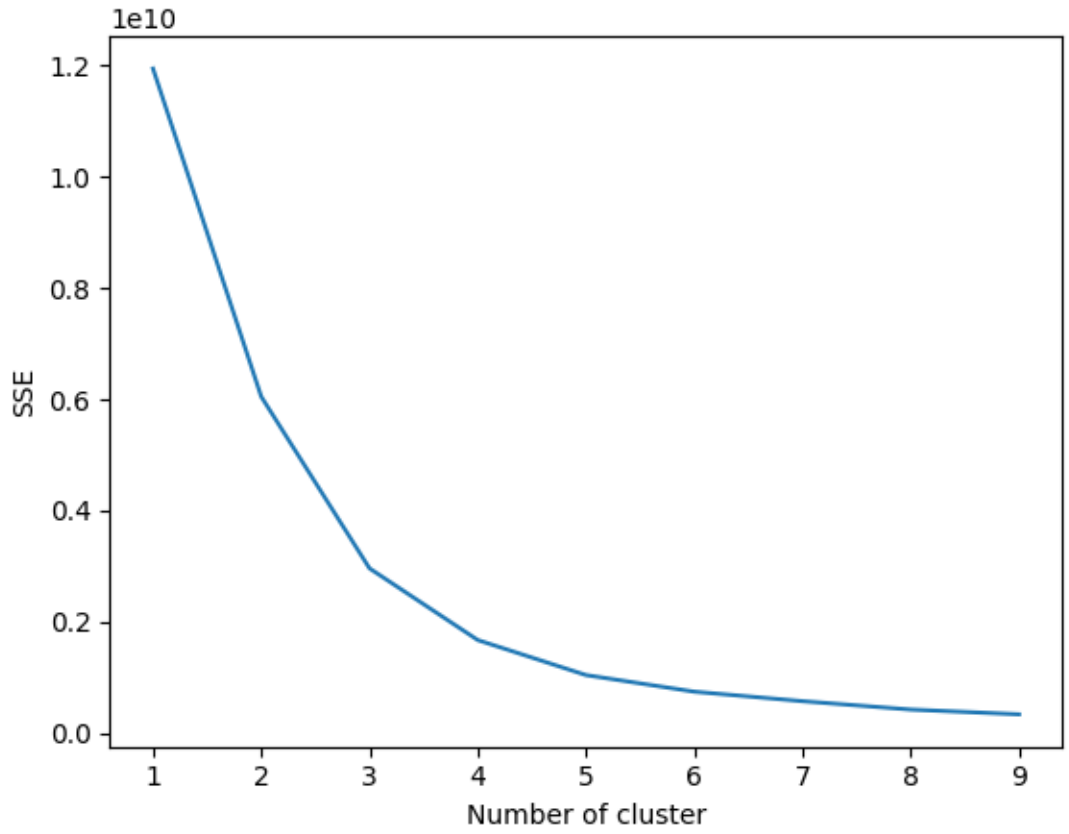


Figure 5: Plot of the inertia of different numbers of clusters to find the elbow point.

From the results we have seen, we can say that the answer to the research question “Can we categorize streamers based on their viewer engagement and streaming activities?” is yes. However, there is more to discuss based on these results. Firstly, the number of clusters we chose was derived from the ‘elbow point’ in figure 5, which made sure we don’t underfit or overfit the number of clusters in our data. Secondly, from the difference between figures 1 and 2, and figures 3 and 4, we can see that scaling the data made a large difference in how the cluster analysis was calculated. Without scaling, most streamers had features that were close to or near zero, as seen by the diagonal histograms in figure 1. This resulted in Cluster 0 having the lions share of streamers. These were streamers that had a low number of streams, viewers, and interactions. For this reason, we will use the scaled data to find trends.

From figure 4, we see that Cluster 0 held most streamers, who had lower means in their statistics. I would categorize these as part-time streamers, those who stream for fun when they feel like it and attract a few viewers. This would be the average streamer on Twitch. Cluster 1 had double the number of streams of Cluster 0, however they had seven times the watch time. We can see that these streamers are more committed to streaming and have a greater entertainment factor than average streamers. Cluster 2 has similar characteristics to an average streamer but has the highest watch time of any cluster. This means that they have few viewers and interactions, but those interactions last longer than others. Cluster 3 has streamers that have the highest number of streams, users, and interactions of any cluster. They have a watch time that is better than Cluster 1 but worse than Clusters 2 and 3. These are the popular full-time streamers, those who stream as much as possible and most likely do

it as their full-time job. We can see that each cluster has their own pros and cons to being a part of their category.

## Conclusion:

Were you able to answer your hypothesis / research questions? Explain how and why (or why not).

I was able to answer the research question in this project. Using K-means clustering algorithm on features that I had derived from the data, I was able to find distinct clusters of streamers that had similar behaviors and interactions with their viewers, allowing categorization of these different clusters. We can see from the pair plots and statistics that these clusters exist and are distinct from each other across the various features that each cluster was calculated from.

What implications do your results have?

The results of this project have several implications. The identified clusters or categories allow streamers to have insights into the different patterns that attract and retain viewers on Twitch. Understanding this could allow streamers to improve their streaming behaviors, such as scheduling, content, and interactions with viewers, resulting in wider audiences and better watch times on their streams. These clusters can also be used to help recommendation systems for viewers, which would suggest streamers to view based on their engagement patterns.

What future questions or directions would you take with your project?

If I were to improve upon this project, I would probably collect more information that could be relevant to the analysis, such as information of what goes on in each streamer's stream, data from different streaming platforms other than Twitch, time-series analysis of streamers over the years, etc. This would allow for a stronger analysis and categorization of the streamers and improve the prospect of a recommendation system for users. This would allow for a better understanding of the streaming environment and would be good for stakeholders to leverage for a more optimized experience.

## Critique of Design and Project:

Although the data visualization aspect of this project gave us some valuable insights into the clusters and categories of streamers, there was room for improvement in terms of showing a better visualization to make it easier to communicate in the analysis. I would have enjoyed creating an interactive plot that would allow me to show the depth and complexity of the clusters. In the pair plots I have created; it is often difficult to see the difference between each cluster as there are crossovers in each subplot. The fact that they are pairwise plots does not give a full view of the higher-order relationships that create the clusters.

Although the scaled data provided better insights into the hidden patterns within the data, it still did not do the data justice. Due to the large outliers in our data, such as streamers like Ninja, Tfue, shroud, etc, there was a large gulf between most streamers and these heavyweights. As there was a clear cluster of streamers like this who outperformed most, perhaps removing these from the dataset and reperforming the cluster analysis would allow

for better insights on the lower end of streamers in the data. This would have made the analysis and visualization much more clear for these datapoints.

## Reflection:

Originally, I had a different project proposal for the RateBeer dataset, however after attempt after attempt I realized that the Json format of the data was not viable for loading and using the data for the project. This meant I had to change my entire project to a different one, which was stressful, however I was helped with some core course teachings. I found that Dask was very useful in this project, as it helped improve the computability of the large datasets and allowed for easy processing of the data into a useful format for my research. The clustering algorithm was perfect to help me formulate and answer the research question. SciKit learn was also helpful as it allowed me to apply machine learning concepts to the big data processes. Stack Overflow was useful in helping me understand and fix bugs in my code. Also, the documentation of the various libraries that I used were helpful in understanding what I was doing throughout the project.

## References:


Recommendation on Live-Streaming Platforms: Dynamic Availability and Repeat Consumption. Jérémie Rappaz, Julian McAuley and Karl Aberer. RecSys, 2021

McAuley, J. (n.d.). *Recommender Systems datasets*.

https://cseweb.ucsd.edu/~jmcauley/datasets.html#twitch

*K-Means Clustering | Brilliant Math & Science Wiki*. (n.d.). https://brilliant.org/wiki/k-means-clustering/

*scikit-learn: machine learning in Python — scikit-learn 1.5.0 documentation*. (n.d.). https://scikit-learn.org/stable/index.html

*pandas documentation — pandas 2.2.2 documentation*. (n.d.). https://pandas.pydata.org/pandas-docs/stable/index.html

Satishgunjal. (2020, July 7). *Tutorial: K-Means Clustering*. Kaggle.

https://www.kaggle.com/code/satishgunjal/tutorial-k-means-clustering

*Dask — Dask documentation*. (n.d.). https://docs.dask.org/en/stable/

*Mining of massive datasets*. (n.d.). http://www.mmds.org/