Prediction of Top Categories on Twitch

I analyzed data on the 200 most watched categories on Twitch each month from January 2016 to October 2020. I found that dataset on https://www.kaggle.com/rankirsh/evolution-of-top-games-on-twitch. That data was originally obtained from https://sullygnome.com/ which has a lot of Twitch statistics available. Each row of the data set contain a Twitch category name, the month and year along with the hours watched, hours streamed, peak viewers, peak channels, total streamers, average viewers, average streamers, and average viewer to streamer ratio for that category that month.

The questions I was interested in answering were:

- How do k-means clusters look for this data? Do they vary year-to-year?
- Can the most popular Twitch categories for next month be predicted from information about this month?

Being able to predict popular Twitch categories could be useful to either further promote those categories or advertising.

Data Cleansing and Preparation

One of the rows in the data appeared to be missing a category name, however; when looking at the original data on sullygnome that category has a Korean name that doesn't copy correctly so I left it in with a blank name.

The dataset also contained information about total hours watched on Twitch for each month in the same time period. Summing up the hours watched for the top 200 games exceeded the reported total hours watched in a few months (April 2016 is one such example). I tried to figure out why this might be, but was unsuccessful so this information was ultimately ignored.

To prepare the data for analysis I created a few extra features. First, I combined the Month and Year features together to count months from January 2016 and called it Month_Since_Jan_2016; this was used for making plotting easier. Additionally I created features based on whether or not a category was in the Top 10, 20, 50, and 100 most viewed the following month, these were the target features for the models to predict.

Data Exploration

I used k-means to attempt to cluster the data. The data was separated by year before doing this. The results are difficult to visualize, but the best looking view I could find was to plot Rank vs Months Since January 2016 and color by cluster, this is shown in Figure 1.

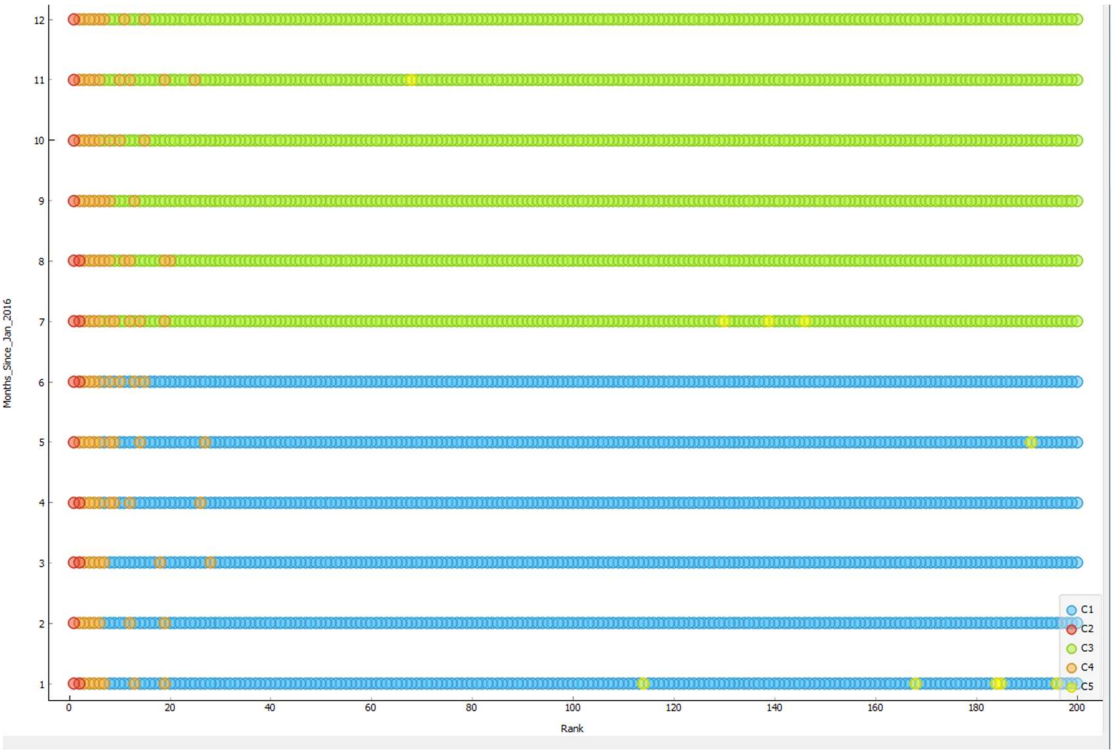

Figure 1: Rank vs Months Since January 2016 for the year 2016, colored by k-means cluster.

The clusters seem to be forming some vertical bands on the left side of the graph, with a split between the first and second half of year between the green and blue cluster, and the yellow cluster just seems to have random points in a few spots. The k-means clusters for other years look similar. Figure 2 shows part of the silhouette scores, there are too many data points to show it all in one picture. Since so many points have large negative scores theses clusters are not very good. I also tried hierarchical clustering; it formed 13 clusters that were almost exclusively by month which isn't useful either. Since neither of these clustering methods seemed to yield anything meaningful I didn't pursue them further.

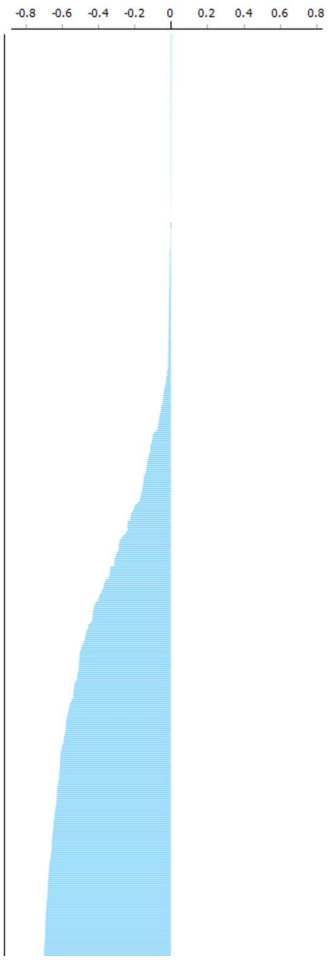

Figure 2: Silhouette scores for k-means cluster of 2016 data

There are many games that have consistently been popular on Twitch

- League of Legends in the top 3 most popular categories every month of this data.
- Fortnite was first in the top 10 in October of 2017 and has been there every month since.
- World of Warcraft has been in the top 15 every month of the data

In the next section we'll see more that popular categories on Twitch tend to stay popular.

Modeling

The main thing I wanted to predict was how popular a category would be on Twitch next month based on its popularity and other statistics from the previous month. For this I used the Top_X_Next_Month (X = 10, 20, 50, or 100) features generated earlier, which are 1 if that category is ranked X or below the following month and 0 otherwise. These are the target features for the models used after this point.

First, I created a simple baseline model for comparison against future models. This model is intended to be as simple as possible and other models are only considered useful if they are significantly better than this baseline. The models I considered for this were a tree and a logistics regression; these ultimately end up being very similar. Both of these models were given just the rank from this month as the only input feature and Top_10_Next_Month as the target feature. The tree was also limited to a depth of one. The results of this are shown in figure 3. The two models are identical in the measures shown except for AUC, so I looked at the ROC curve, shown in Figure 4, to see why this would be. The biggest feature to notice about these curves is that the one for the tree is just two lines while the one for the logistics regression is a smoother curve. From my understanding, this is because the logistics regression is more granular with its prediction probabilities while the tree is assigning everything with rank <= 9 this month with one probability and everything else a second probability (this can be seen in the tree viewer). So the models are making the same predictions, if we set a threshold of .5, but the logistics regression has a smother ROC curve leading to a higher AUC.

Evaluation Results

Model	AUC	CA	F1	Precision	Recall
Tree	0.912	0.987	0.859	0.891	0.829
Logistic Regression	0.988	0.987	0.859	0.891	0.829

Figure 3: Comparison of Tree and Logistic Regression on predicting Top 10 Next Month

Continuing from here, I used tree models as the baseline because its results are easy to read from the tree viewer, but I will be ignoring AUC in comparing to other models since AUC seems to not be a great measure of the quality of the predictions of the tree in this case.

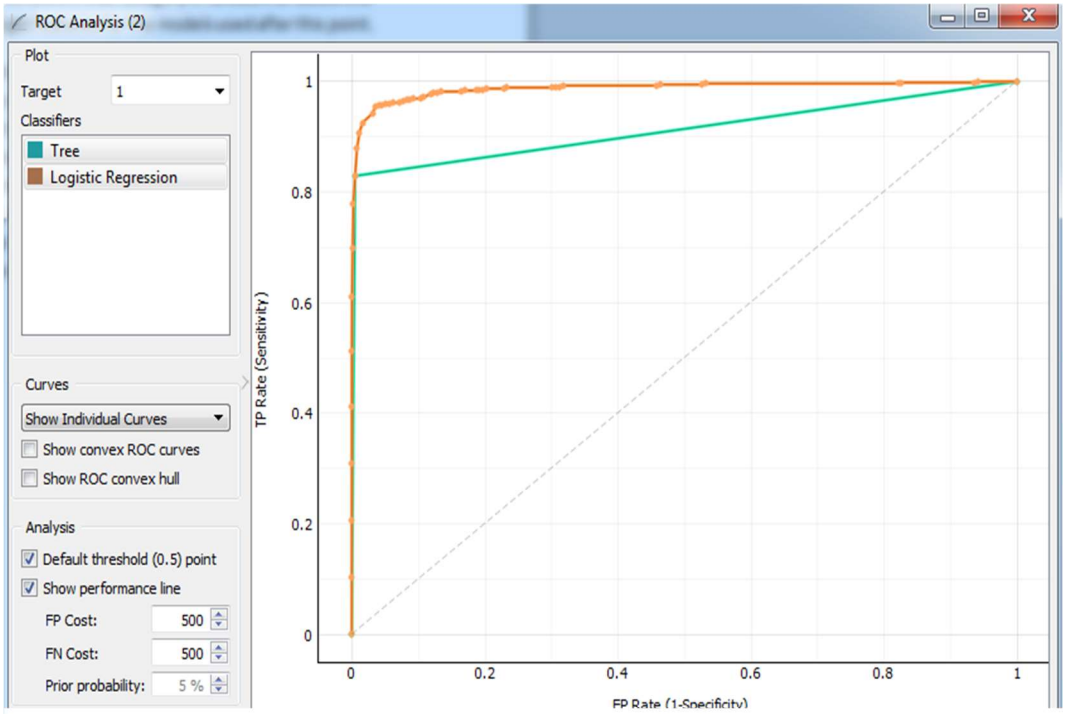

Figure 4: Comparison of Tree and Logistic Regression ROC curves on predicting Top 10 Next Month

The next step is to compare this baseline to some other models. The models I chose were a neural network, a logistics regression, and naïve Bayes. These models were given all the available features as input and one of the Top_X_Next_Month features as a target. They were also given 70% of the dataset as training data and the remaining 30% as test data; this is done to reduce the possibility of overtraining the models. The results from these models are shown in figure 5. Comparing these numbers to Figure 3, ignoring AUC, we see that these models are actually worse in most measures. Naïve Bayes has better recall than the simple tree, but much worse in other measures especially precision.

I repeated this procedure, trained a simple tree then compared it to the other models, for the each of the other target features. The results of this are shown in figures 6, 7, and 8. In all 4 cases we can see that the simple tree created outperforms the other models (ignored AUC).

Recommendations:

Since the simple trees performed so well compared to the other models it best to just use them for predicting what the Top 10, 20, 50, 100 Twitch categories will be for the next month. Each tree predicts that anything with a rank less than or equal to a given cutoff will most likely be in the Top X next month and anything above that cutoff will most likely not be in the Top X next month. Those cutoffs are 9, 20, 49, and 93 for the Top 10, 20, 50, and 100 respectively.

There might be a way to improve these predictions with more sophisticated models or with more data, but because a simple comparison test seems to perform so well it's hard to justify spending much additional time or resources on this.

A better use of resources might be to try to how popular games that are about to be released, such as Cyberpunk 2077, will be on Twitch.

Evaluation Results

Model	AUC	CA	F1	Precision	Recall
Neural Network	0.987	0.985	0.824	0.832	0.816
Logistic Regression	0.985	0.984	0.804	0.858	0.757
Naive Bayes	0.968	0.876	0.408	0.259	0.961

Figure 5: Comparison of Neural Network, Logistic Regression, and Naive Bayes on predicting Top 10 Next Month

Evaluation Results

Model	AUC	CA	F1	Precision	Recall
Neural Network	0.966	0.961	0.961	0.960	0.961
Logistic Regression	0.969	0.955	0.952	0.953	0.955
Naive Bayes	0.953	0.883	0.900	0.938	0.883

Evaluation Results

Model	AUC	CA	F1	Precision	Recall
Tree	0.911	0.965	0.966	0.966	0.965

Figure 6: Comparison of Neural Network, Logistic Regression, and Naive Bayes on predicting Top 20 Next Month

Evaluation Results

Model	AUC	CA	F1	Precision	Recall
Neural Network	0.943	0.923	0.922	0.922	0.923
Logistic Regression	0.940	0.911	0.906	0.910	0.911
Naive Bayes	0.938	0.880	0.884	0.894	0.880

Evaluation Results

Model	AUC	CA	F1	Precision	Recall
Tree	0.911	0.929	0.929	0.931	0.929

Figure 7: Comparison of Neural Network, Logistic Regression, and Naive Bayes on predicting Top 50 Next Month

Evaluation Results

Model	AUC	CA	F1	Precision	Recall
Neural Network	0.921	0.857	0.857	0.857	0.857
Naive Bayes	0.897	0.818	0.816	0.818	0.818
Logistic Regression	0.904	0.818	0.813	0.832	0.818

Evaluation Results

Model	AUC	CA	F1	Precision	Recall
Tree	0.858	0.857	0.857	0.859	0.857

Figure 8: Comparison of Neural Network, Logistic Regression, and Naive Bayes on predicting Top 100 Next Month