Cindy Hernandez
CIS 3120
Professor Jairam
14 May 2024

Classifying Twitch User Metrics

Twitch is one of the leading live-stream services that gains large amounts of data each day that can be used as insight for improving streaming content, user behavior and peak viewership, as well as streaming dynamics. Analyzing Twitch data using classifiers offers several strong and effective approaches to understanding viewer metrics and gaming trends within the platform. By utilizing machine learning classifiers such as Naive Bayes, Support Vector Machines, K-Nearest Neighbors, Decision Trees, and Random Forests, we can categorize and predict various aspects of Twitch activity. These classifiers can help in categorize viewers based on their watch time and peak hours, identifying key factors that drive engagement, and improving content information. These analytical approaches can not only enhance overall user experience but could also aid streamers and platform developers in making data-driven decisions to increase revenue, boost interactions and satisfaction on Twitch.

The process of building any classification model involves several important steps: data preprocessing, feature selection, data transformation, model training, and evaluation. To begin preprocessing the data, I loaded the data set into CSV file that I found via Kaggle. After loading the dataset into a pandas Data Frame, I began removing columns that were not needed for the analysis. Such columns were 'Peak_channels', 'Rank', 'Year', 'Month', 'Avg_viewer_ratio', and 'Game'. Any rows with missing values were removed from the dataset to ensure that there would be no errors within the calculation process.

Since the variable ‘Hours_watched’ was a continuous variable, I decided to categorize it into bins and labels to group the data into discrete variables. The bins used were [80000, 70000000, 139920000, 209840000, 279760000, 349680000], and the labels assigned to these bins were [1, 2, 3, 4, 5]. This binning process was crucial to the analysis because it allowed is to categorize these variables that were essential to our analysis. Had I kept them the same way they were found in the dataset, there would have been a calculation error that would prevent us from creating a understandable analysis. I then used an ordinal encoder to transform these categorical labels into numerical values. The original column was named 'Hours_watched bins' which then encoded to make a new column ‘Hour watched level’, which we would then use as the target variable for our model. After encoding it, I was able to drop both 'Hours_watched' and 'Hours_watched bins' columns, leaving the dataset with only the necessary features for analysis.

The next step was to prepare the features and target variable for model training. The feature set (X) included all the columns except 'Hour watched level'. The ‘Hour watched level’ would be used as our target variable (y). To standardize our features, a StandardScaler was applied to transform the feature set. Standardization is crucial for classifiers such as K-Nearest Neighbors, Support Vector Machines and Naïve Bayes, as it ensures that all features contribute equally to the model's performance without having high values that would conflict with the model results. The dataset was then split into training and testing sets using the train_test_split function, using the 70:30 ratio, meaning 30% of the data would be used for testing and 70% for training. This split ensures that the model could be evaluated on unseen data to assess its performance and promotes reliability.

The Gaussian Naive Bayes classifier was selected first for the analysis. This type of Naive Bayes model assumes that the features are following a normal distribution. The model was trained using our training dataset. The fit method was used to train the model on the standardized feature set (X_train) and the target variable (y_train). Once the model was trained, it was evaluated using the testing dataset. Predictions were made on the test set (X_test) using the predict method. The accuracy of the model was calculated by comparing the predicted values to the actual target values (y_test). After completing the training and testing dataset analysis, I printed out a classification report which would print out the accuracy score and include information regarding the precision, recall, and F1-score for each of the classes tested. The classification report is crucial for insight into the model's performance across the different categories of the target variable.

For the Naive Bayes classification report, it produced several resulting within the classes. Precision, which measures the proportion of true positive predictions among all positive predictions, displayed that the model was relatively accurate in predicting certain categories. The recall score measures the proportion of true positive predictions among all actual positives. Within the classes, class 1, 2, and 3 was able to identify most instances whereas class 4 was less accurate in finding these instances. The F1-score, which is the combined mean of precision and recall, provides a single metric that balances both precision and recall and is useful when classes are uneven. In our classification report, the F1-score for most classes were effective, however, class 1 and 4 has smallest F1-score which would need some improvement.

In addition to the Naive Bayes classifier, four other classifiers were used in this analysis including Support Vector Machines (SVM), K-Nearest Neighbors (KNN), Decision Trees, and Random Forest. Each of these classifiers has unique approach, as well as characteristics that provided different results on their classification reports. The Support Vector Machine (SVM) classifier, known to be effective in classification tasks and regression, performed well with a high accuracy rate, however the individual class scores were not strong and were subpar to some of the other analyses. The K-Nearest Neighbors (KNN) classifier, which classifies data points based on the majority class among its nearest neighbors, yielded good results, but would not be as useful in this application because of the expensive computation for large datasets.

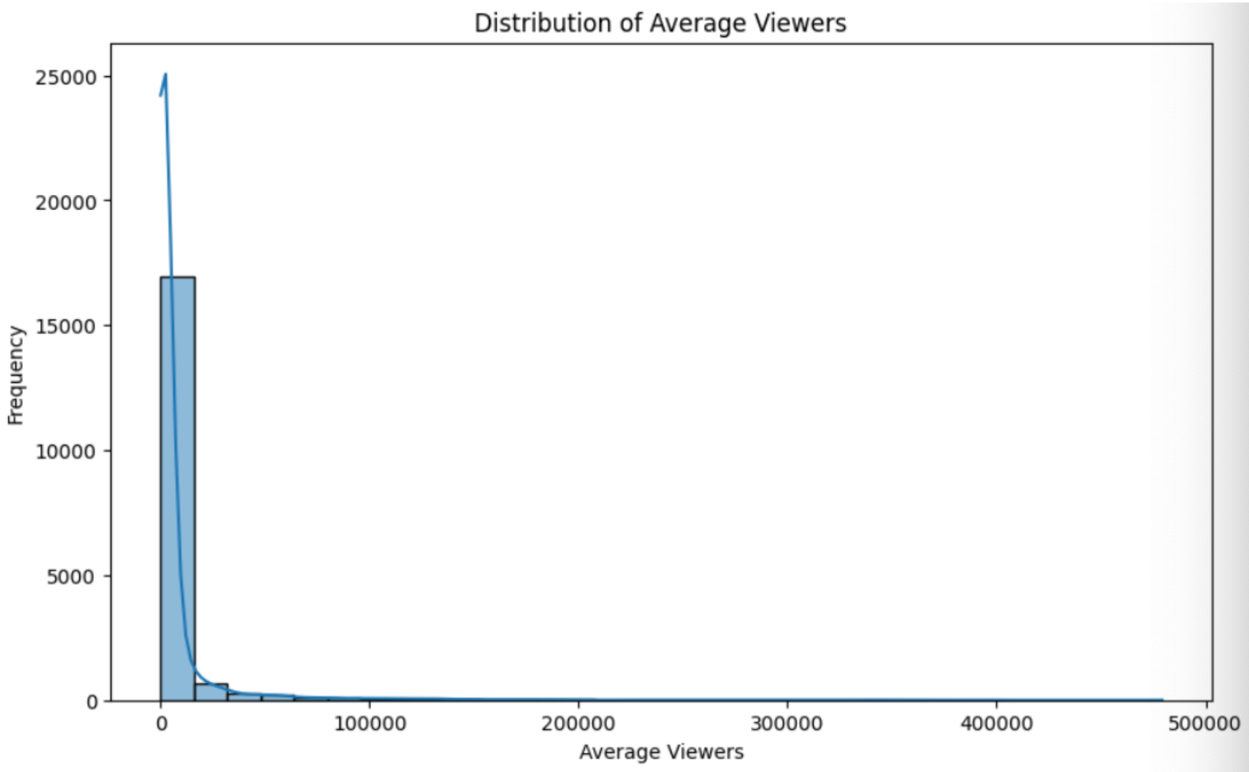
The Decision Tree classifier, which splits the data into branches to make decisions, offering intuitive and easy-to-interpret results. It performed very well on the dataset, but it may be overfitting as the test data was a bit lower in class 4. The Random Forest classifier, an ensemble method that combines multiple decision trees fixes this issue by averaging the results of several trees. Its classification report was high and accurate, displaying that it was very successful with this dataset. Each of these classifiers contributed valuable perspectives on the dataset in different approaches, showcasing their ability to look at multiple analyses In the context of the Twitch game data classification.

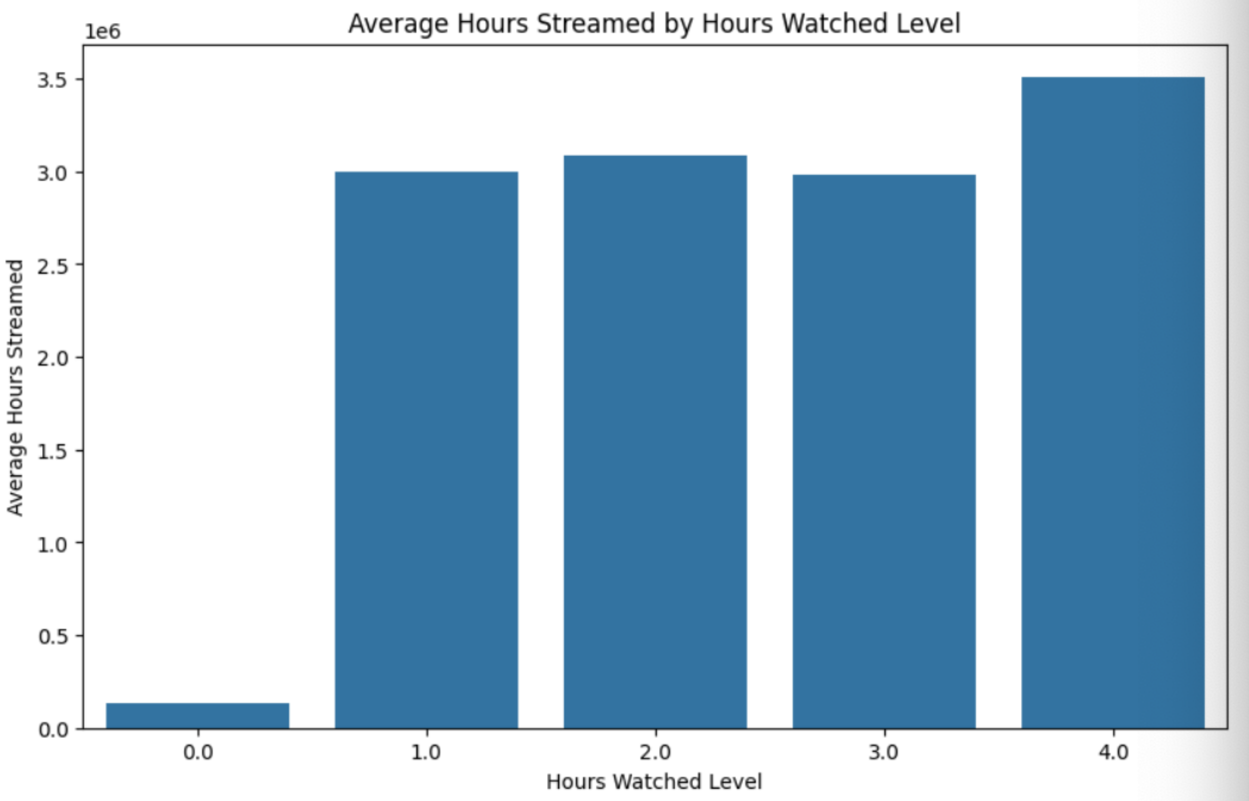
Overall, the classification report revealed that the Naive Bayes, SVM, and KNN models performed well in certain areas but had room for improvement in others. Despite Decision Trees having some good results, the overfitting of the model may become an issue that will not give us the analyses needed for our classification. Random Trees performing well was surprising since I was expecting it to be the least successful and useful. The detailed metrics provided valuable insights into the model's strengths and weaknesses and guiding us further refinements and adjustments to enhance its predictive capabilities.

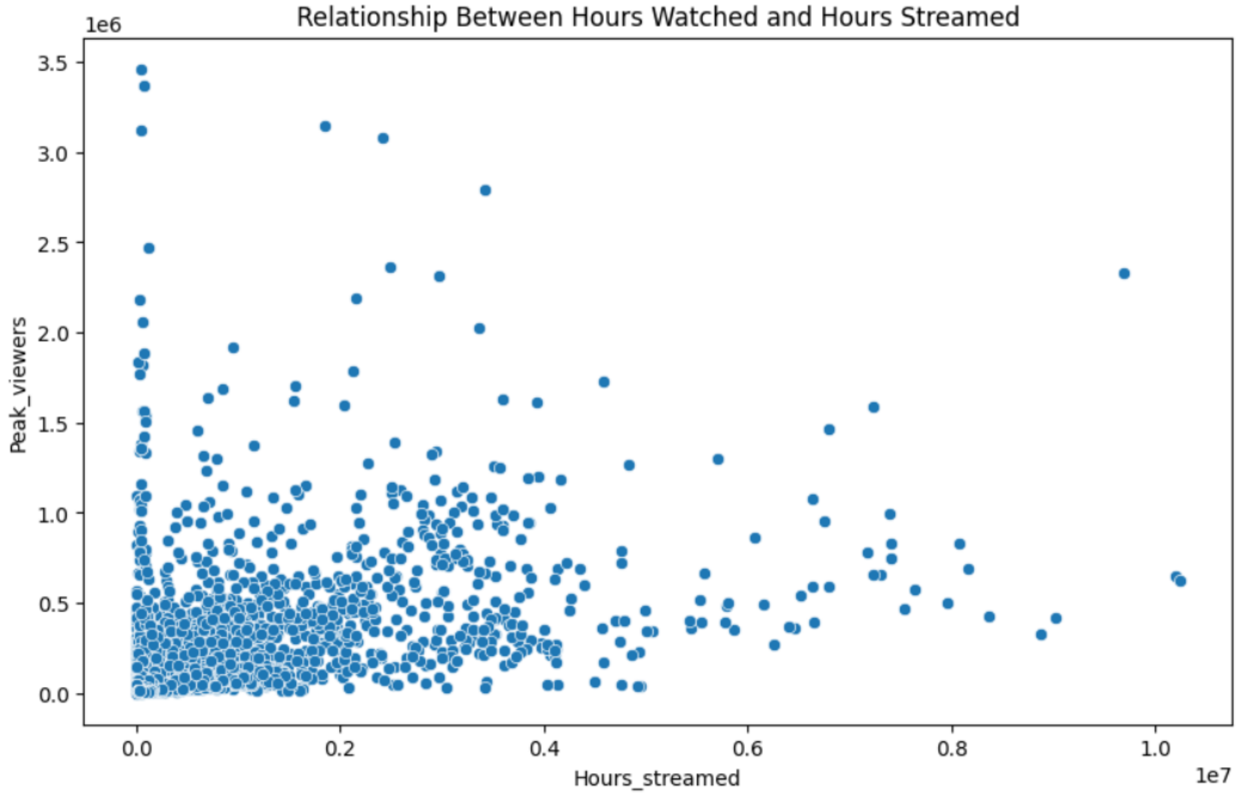
To further understand the relationship between the variables, a scatter plot was created using seaborn. The plot visualized the relationship between 'Hours_streamed' and 'Hours_watched', providing a clear plot that showcases the data distribution and potential correlations between these two features. I also created a histogram to show the distribution of Average viewers. From the visualization, we can see that most of the data is skewed to the right, which contains the smaller values of average viewers. This displays the number of average viewers and their frequency. Lastly, I created a bar graph to display the hours watched based off the bins we created earlier. This showcases the hour watched level by the hours streamed, showcasing a nice category for the watch bins.

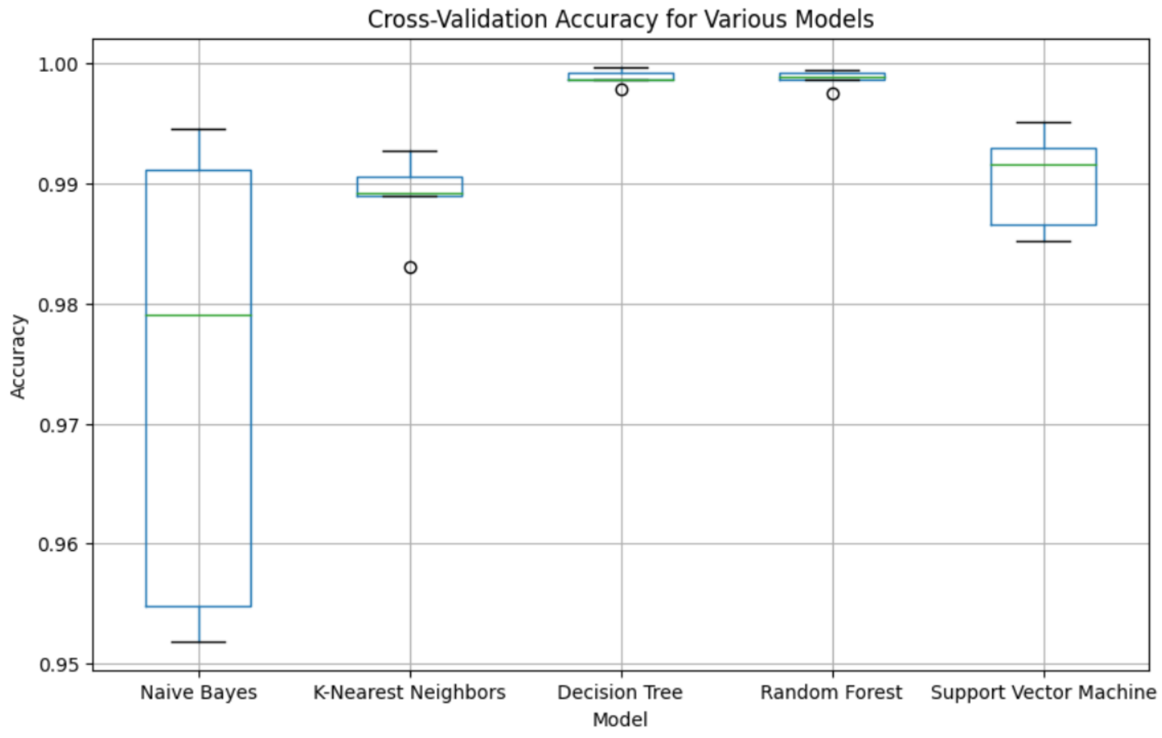
One of the challenges I faced during this project was creating bins to categorize the hours watched. When first conducting this analysis, I was receiving errors and inaccurate accuracy scores because the values were too big and unique to provide a good analysis. To solve this issue, the bins were used to categorize the values and encode them. Then those encoded values were used for the analysis and solved the issues I had when executing the model. Creating the bin sizes was also difficult for me since I wasn’t sure what numbers to use to categorize them properly. After looking at the min and max values and separating them into 6 bins, I was able to categorize them properly.

All in all, the analysis of Twitch metrics through these classifiers showcased the usefulness of machine learning to understand and interpret large amounts of data. These data sets could then be analyzed further and pull-out specific information that can be used for improving user engagement, overall stream content, and increase revenue. Each of the classifiers used within this analysis were effective in their own ways and could be used to view and manage multiple problem that Twitch may encounter.


Distribution of Average Viewers
Frequency
Average Viewers
0
5000
10000
15000
20000
25000
100000
200000
300000
400000
500000


Average Hours Streamed by Hours Watched Level
1e6
Average Hours Streamed
Hours Watched Level
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0.0
1.0
2.0
3.0
4.0


Relationship Between Hours Watched and Hours Streamed
1e6
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Peak_viewers
0.0
0.2
0.4
0.6
0.8
1.0
1e7
Hours_streamed


Cross-Validation Accuracy for Various Models
1.00
0.99
0.98
0.97
0.96
0.95
Accuracy
Naive Bayes
K-Nearest Neighbors
Decision Tree
Random Forest
Support Vector Machine
Model