Twitch Streamer Partnership

by Griffin Hundley

Overview

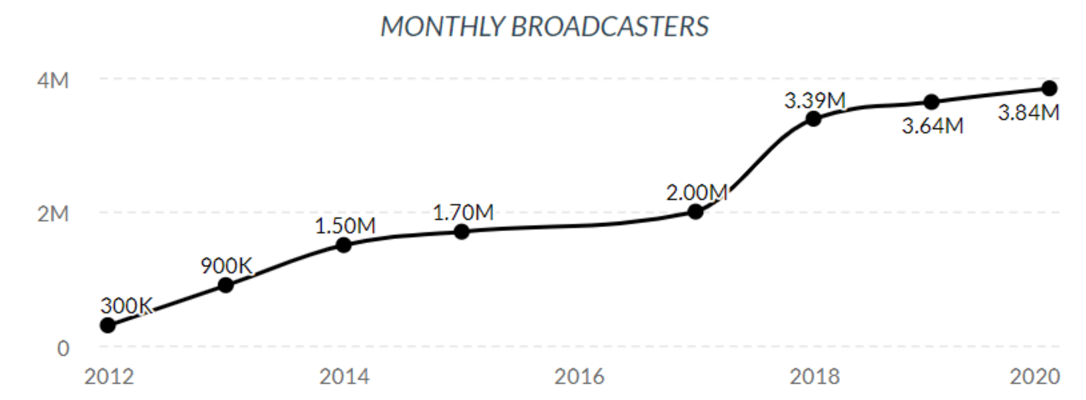
Twitch is a social platform for livestreaming video games, music, art, and more. For most people making content on the platform, livestreaming started out as a hobby for people sharing their love of gaming - something they do only because its fun. For a select few who build a strong community and gain a larger audience, they can turn their channel into a primary source of income by becoming a partner with Twitch.

Being a partner has a lot of perks, but also a few restrictions. Partners get a subscription button added to their channel, where viewers can pay $5 for 1 month access to subscriber-only emotes, don't have to watch advertisements, and receive channel specific rewards. Partners also get a share of the revenue for ads ran on their channel. As part of the contract that partners sign, they cannot livestream on other platforms.

As of February 2020, there are 3.8 million unique broadcasters on the website, with an average of 56,000 concurrent broadcasters, and 1.44 million concurrent viewers at a given time. Of those 3.8 million broadcasters, there are roughly 41,000 partners.

Source: Mansoor Iqbal https://www.businessofapps.com/data/twitch-statistics/

Business Problem



Source: Mansoor Iqbal https://www.businessofapps.com/data/twitch-statistics/

Livestreaming is emerging as a premiere online industry, with significant growth over the years. As of December 2020, the effects of COVID are still being realized, however there are reports of 24-50% increase in Twitch viewership as a result of the pandemic lockdowns.

Online entertainment isn't going anywhere but up. As a result of increasing scale, more online companies must rely on automated systems. For example - Twitch competitors Youtube Live and Facebook Gaming use automated systems to detect copywrite infringment and obscene content. As the number of broadcasters and viewers continues to rise, and as the entertainment industry shifts more online, the number of people turning to livestreaming as a source of income increases.

Applications for partnership are currently reviewed manually by Twitch staff, with applicants waiting 2-4 weeks to get results. Because partnership is approved manually by different individuals, there can be a personal selective bias towards or against certain demographics. In some cases, very successful streamers with well above average metrics have been denied partnership, while much smaller channels are approved, leading to distrust between the broadcaster and Twitch.

The goal of this analysis is to provide a model that can approve or deny channels for partnership based on their channel metrics, removing the need for manual review, which frees up resources, decreases feedback time to streamers on the status of their partnership application, and having a system without personal influences. Additionally, the nature of the model allows for dynamically adjusting the threshold for approval, to make it more or less strict depending on the needs of Twitch.

Source: Thomas Wilde https://www.geekwire.com/2020/twitch-sets-audience-record-october-pandemic-continues-fuel-livestreaming-growth/

Data

Over the course of a week, at different times during the day, the Twitch API was queried through all of the pages of currently live broadcasters to generate a list of ~800,000 broadcasters. These broadcasters were then filtered to only channels older than 60 days, with over 900 viewers (~300,000 broadcasters). This is to filter out brand new channels, and only to examine channels that meet the basic number of viewers to be considered for partnership as described in the Path to Partnership achievement. From the twitch api, the data being collected covers:

`view_count`: total number of lifetime channel views

`account_age`: time since account creation

`broadcaster_type`: (partner, affiliate, unaffiliated)

From these ~300K broadcasters, a random sample of 10% of those channels (~30k) was chosen to obtain a reasonably sized dataset. For these ~30k channels, data was webscraped from twitchtracker.com. The channel metrics being examined are lifetime aggregate channel data. The specific metrics gathered are:

`hours_streamed`: total number of hours stream has been live

`average_viewers`: average number of concurrent viewers

`peak_viewers`: peak number of concurrent viewers

`days_of_activity`: total number of days where stream was live

`total_games_streamed`: total number of games streamed

`daily_broadcast_time`: average hours channel is live per day

`hours_watched_daily`: average number of hours watched by viewers per day

`followers_per_stream`: average number of followers gained per stream

`views_per_stream`: average number of concurrent viewers per stream

`followers_per_hour`: average number of followers per hour

`views_per_hour`: average number of views per hour

`hours_watched`: total hours watched by viewers

`active_days_per_week`: average of how many days per week the broadcast is live

`average_games`: average number of games played per stream

Data Assumptions

A key assumption being made with this analysis is that a channel's total lifetime aggregate metrics can be used to classify a channel by partnership. Not having access to the date when a channel is partnered, analysis can only be made on the metrics as they currently are, not when the channel was partnered. As a result, theoretically this biased analysis should produce a harsher classifier, with the idea that once a channel is partnered it will continue to grow and inflate its metrics, and by association inflate the decision boundary. If a model produced from this analysis were to be used by Twitch in its partnership approval process, in deployment, Twitch would have access to those specific metrics and would not have this bias.

Features not used in this analysis that Twitch currently uses in their partner selection is chat interaction - how active are the viewers in the chat room. This information would theoretically be important to classifying small streams, as a small stream with a very active chat could be partners.

Additionally, Twitch makes a distinction between 'natural' views and views gained through hosts and raids (when one channel sends all of its viewers to another channel, and only natural views count towards their decision. This analysis has no method of identifying natural views from views gained through hosts or raids, and will assume that views are natural. The theoretical effect of this assumption is a slight deflation of importance in view-related features.

Model and Evaluation

The final model used in this analysis, details found in the modeling notebook, is a neural net binary classifier with the following architecture:

1 Input layer (18x1) ->

4 node Dense Layer with 20% Dropout and Relu activation ->

8 node Dense Layer with 20% Dropout and Relu activation ->

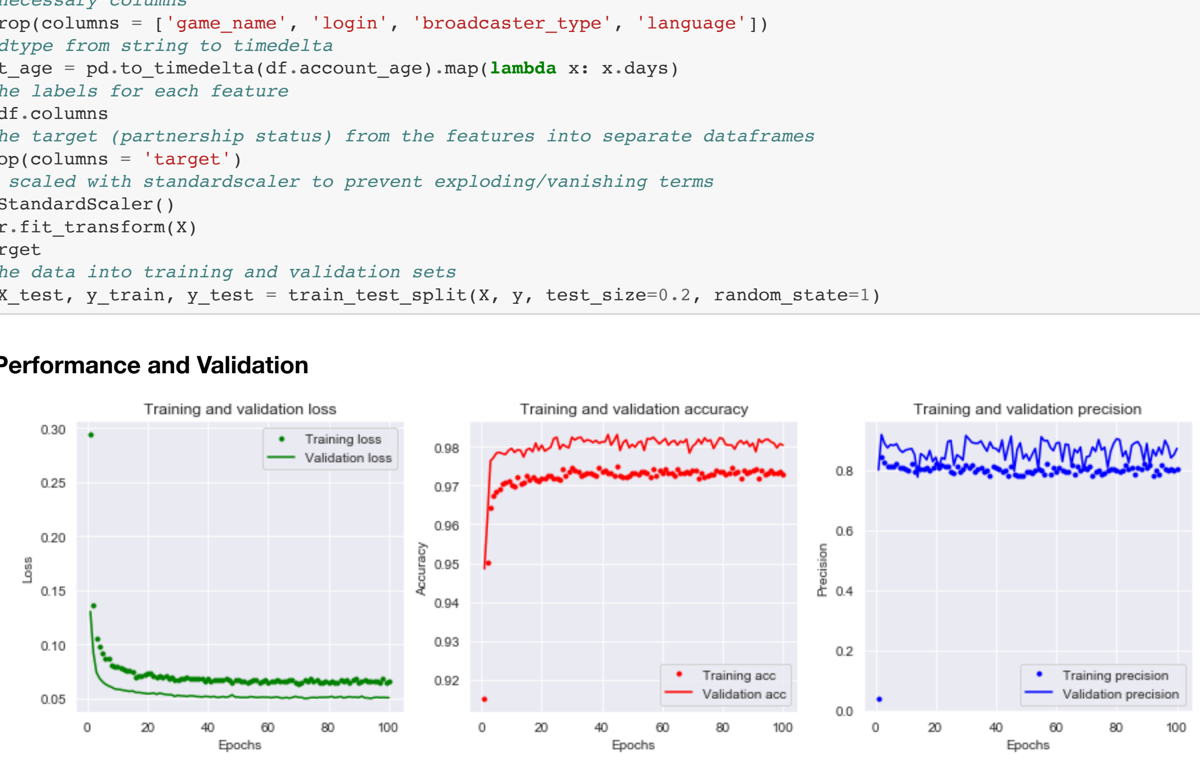
1 Output layer with sigmoid activation

The model was optimized with adaptive moment estimation, used binary crossentropy as the loss function, and the data was fit over 100 epochs with minibatches of size 32.

In [3]:
```
# import the model from the models directory
from keras.models import load_model
model = load_model('../models/model_4.h5')
```

In [4]:
```
# imports
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
# load in the data
df = pd.read_csv('../data/streamer_data.csv')
# drop unnecessary columns
df = df.drop(columns = ['game_name', 'login', 'broadcaster_type', 'language'])
# change dtype from string to timedelta
df.account_age = pd.to_timedelta(df.account_age).map(lambda x: x.days)
# store the labels for each feature
labels = df.columns
# split the target (partnership status) from the features into separate dataframes
X = df.drop(columns = 'target')
# data is scaled with standardscaler to prevent exploding/vanishing terms
scaler = StandardScaler()
X = scaler.fit_transform(X)
y = df.target
# split the data into training and validation sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
```

Model 4 Performance and Validation



In the figures above, this model performs the best of all the iterations. The reduction in complexity, as well as the dropout regularization, has the model consistently perform better on the validation set than on the training set indicating high generalizability. Additionally, the extreme amounts of fluctuations over epochs seems to have been smoothed out considerably.

In [5]:
```
score = model.evaluate(X_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
print('Test precision:', score[2])
print('Test recall:', score[3])
```

```
Test loss: 0.05081172659993172
Test accuracy: 0.9803656339645386
Test precision: 0.8715953230857849
Test recall: 0.7296416759490967
```

Examining the scores by threshold shows a plateau in the f1/recall score that sharply drops off at around 0.75. At that threshhold the model achieves a precision of ~0.95.

In [10]:
```
from sklearn.metrics import f1_score, precision_score, recall_score, confusion_matrix
threshold = 0.74
y_preds = model.predict(X_test)
y_preds_thresh = [1 if x > threshold else 0 for x in y_preds]
f1 = f1_score(y_test, y_preds_thresh)
prec = precision_score(y_test, y_preds_thresh)
recall = recall_score(y_test, y_preds_thresh)
print('Test F1:', f1)
print('Test precision:', prec)
print('Test recall:', recall)
```

```
Test F1: 0.7370517928286853
Test precision: 0.9487179487179487
Test recall: 0.6026058631921825
```

In [11]:
```
tn, fp, fn, tp = confusion_matrix(y_test, y_preds_thresh).ravel()
specificity = tn / (tn + fp)
print(specificity)
```

```
0.9982146045349045
```

Using a confusion matrix on the test set to calculate the specificity gives a result of 99%. With a precision of 95%, this model succeeds in being very selective against non-partners.

Conclusion

This model performs very well at being selective. Although it suffers a bit with a 60% hit rate on partnered channels, because denied applications can reapply at a later point, its not a permanent decision, where in the reverse case, giving a stream partnership is usually permanent. Twitch currently approves ~5k partners per year. Assuming they accept 5% of applications that would mean they receieve 100K applications per year, spending 2-4 weeks reviewing each one. This model would greatly reduce the amount staff resources. In addition, because it is a model based on continuous data, it does not have the same personal biases that humans do, leading to a fairer assessment of partners.

Next Steps

Moving forward I would like to address the assumptions made with the data. If I had access to Twitch's data that excludes artifical views, and can target historical data at the time of partnership, the model would likely be much more accurate at predicting the partner class.

Additionally, expanding the amount of data collection is a logical next step. In this analysis only 4% of the total partners were included in the dataset.

Twitch currently uses chat activity as an important metric when deciding partnership. Getting data on chat participation and activity would thus be a good next step to take.

I didn't have time in this analysis to get data from Twitch on how many partnership applications they receive.

In [ ]: