



**Predicating Successful Clips**



# What's Twitch?

Social video platform for gamers, video game culture, and the creative arts.



**10mm** Daily Unique Viewers



**106 Minutes** Viewed daily per user



**9Bn +** Chat messages in 2015



**35K** Twitch Con 2016 attendees



twitch



# What are Clips?

**twitch**

Clipped by KappaCares

Introducing Clips X

Capture your favorite moments with a single click! Hit the Clips button below and be the first to start sharing.

[Learn More](#)

...

...

...

...

Twitch Weekly - LIVE every Friday at 1pm PT!  
Twitch playing Gaming Talk Shows

1,232 9,521,372 91,568

**twitch**



# Goals

- Predict the number of plays a clip will receive when it is created

## Application:

- Automatically create Twitch and YouTube VOD content
- Provide editors a better starting place to create highlight reels
- Give content marketing an easy place to find good content
- Feature importance:
  - Popularity (games, geography, platform, broadcaster)
  - Virality (referral platform, spreading patterns, community mapping)



# Problem Type

**BASIC FORM**

**4**

		continuous	categorical
supervised	regression	classification	
	dimension reduction	clustering	
unsupervised			



# Initial Data

```
In [6]: #check nulls  
df1.isnull().sum()
```

```
Out[6]: minute          0  
country         0  
region        3570  
asn_id          42  
domain          0  
channel          0  
play_session_id 99405  
url              0  
app_version     99405  
channel_id       0  
partner         200  
clip_id          0  
login             0  
platform          0  
device_id        1224  
referrer_url    100000  
game            1023  
host              0  
source_offset    59484  
duration        58818  
user_id         59484  
total_plays      0  
ccus             1  
msg_per_min      0  
avg_subs          0  
total_bits        0  
number_of_bits_msgs 0  
dtype: int64
```

Approximately 50 million clips are created each month

- Randomly sampled 100,000 clips over the past 4 months

## Features Added

ccus	Number of people that were watching the channel live when the clip was created
msg_per_min	Chat velocity
avg_subs	Number of paid subscribers during the month the clip was made (on that channel)
total_bits	Donations (tips) that occurred during the time of the clip being created
number_of_bits_msgs	Number of msg's that contained a donation/tip



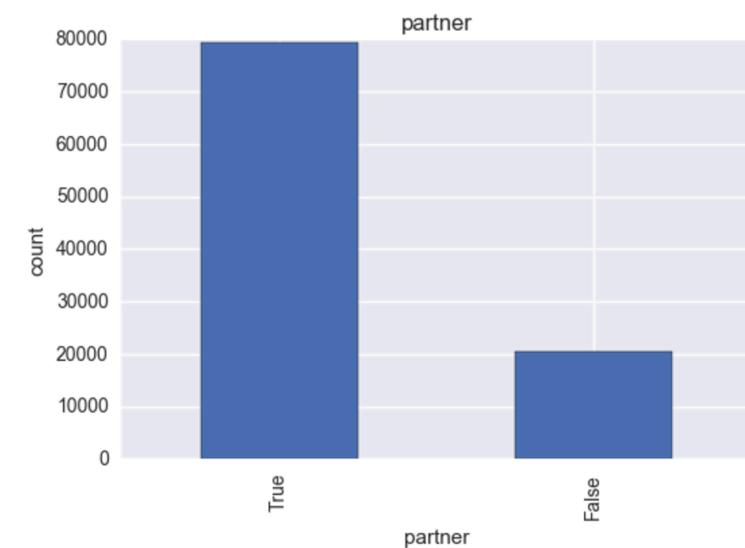
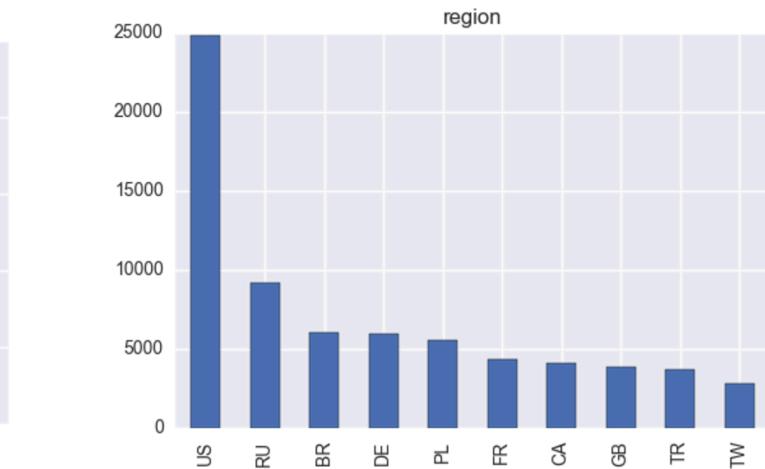
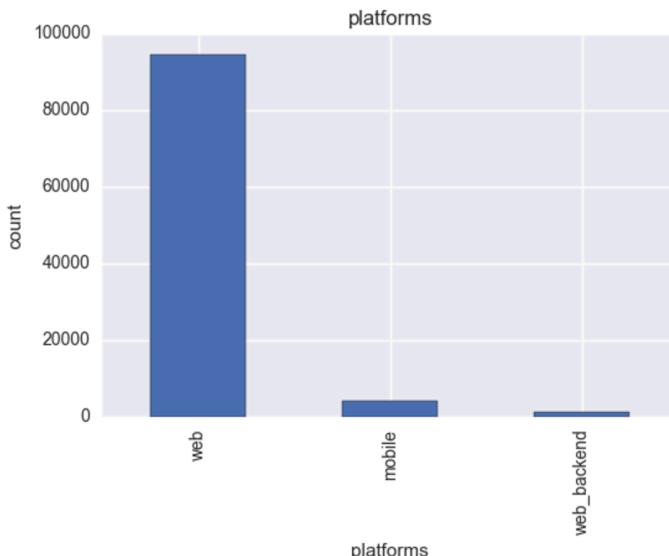
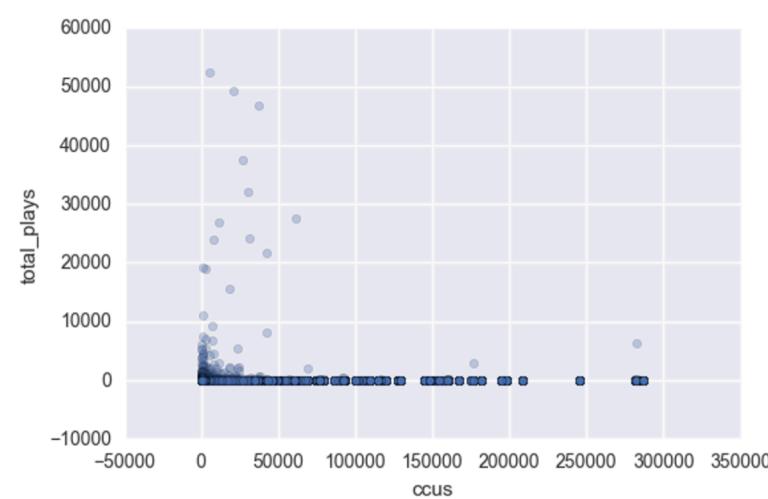
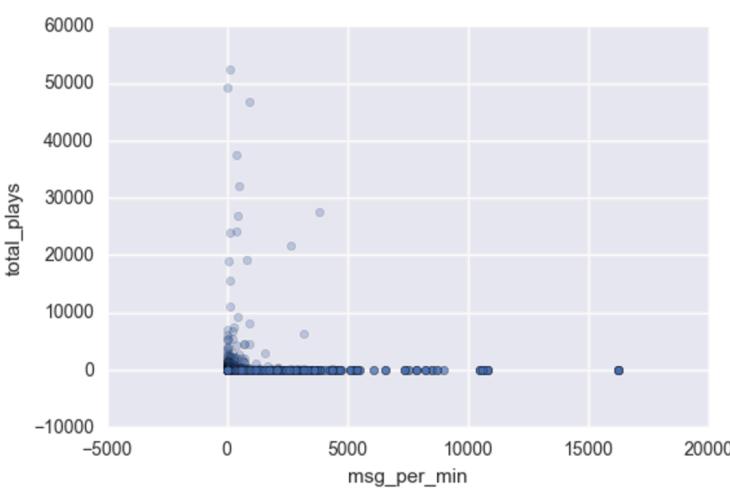


# EDA

```
In [29]: df1.total_plays.describe()
```

```
Out[29]: count    99799.000000
mean      8.015982
std     385.588382
min      0.000000
25%     0.000000
50%     0.000000
75%     1.000000
max    52290.000000
```

Name: total\_plays, dtype: float64





# Results: LinearRegression

Loss Function Used: RMSE

Predicting with the mean of the entire dataset

```
In [23]: np.sqrt(metrics.mean_squared_error(df1.total_plays, df1.total_plays_prediction))  
Out[23]: 385.58644994988981
```

Linear Regression with handpicked features: RMSE = 183.66

- Average score of five cross validations

```
In [63]: feature_cols = ['avg_subs', 'msg_per_min', 'total_bits', 'ccus']  
X = df1[feature_cols]  
cross_val_rmse(X, y)  
  
Out[63]: 183.66020497432285
```

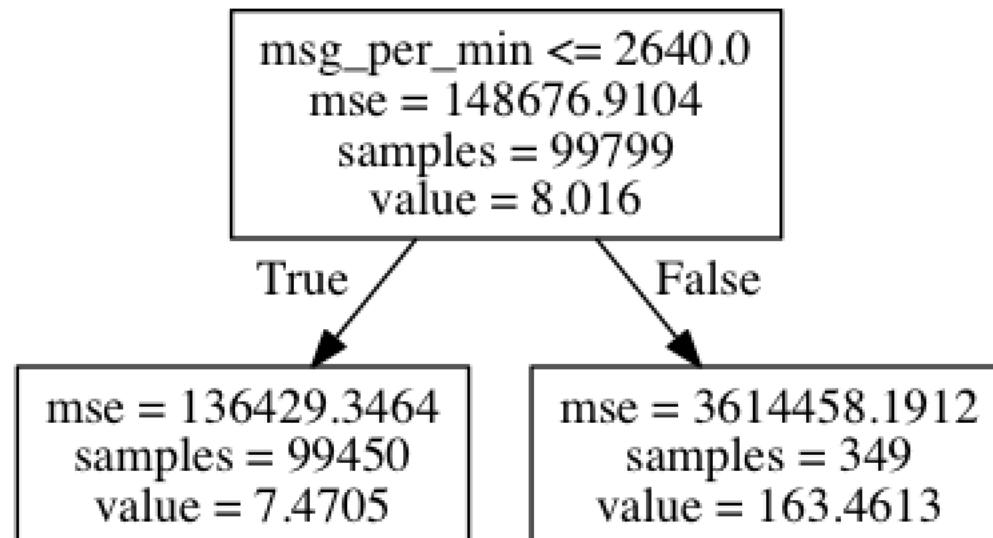


# Results: DecisionTreeRegressor

All features, RMSE: 228.031

Handpicked features & cv, RMSE: 133.86

- Most useful feature was chat velocity (msg\_per\_min)



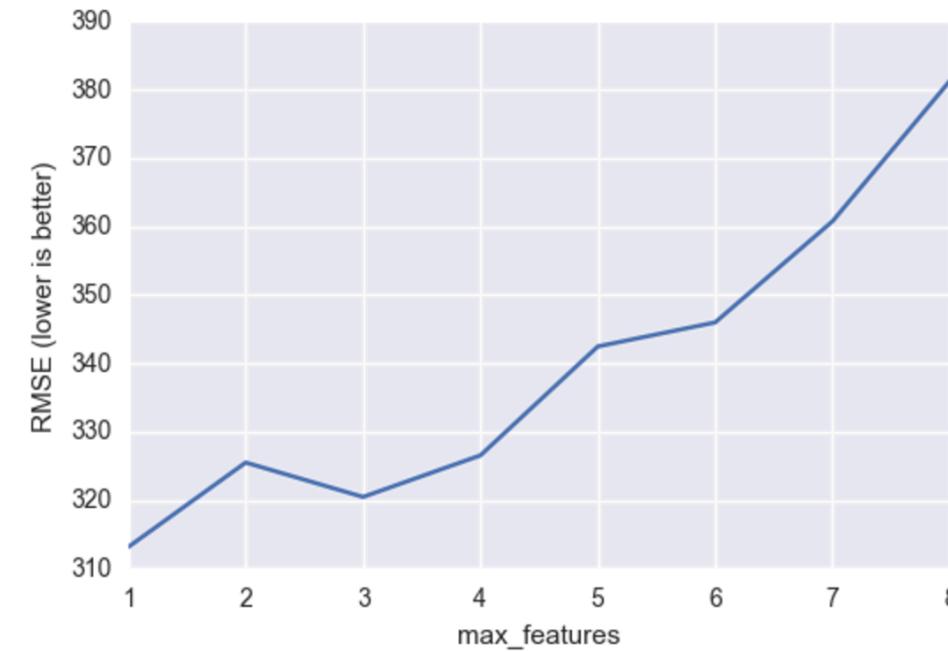
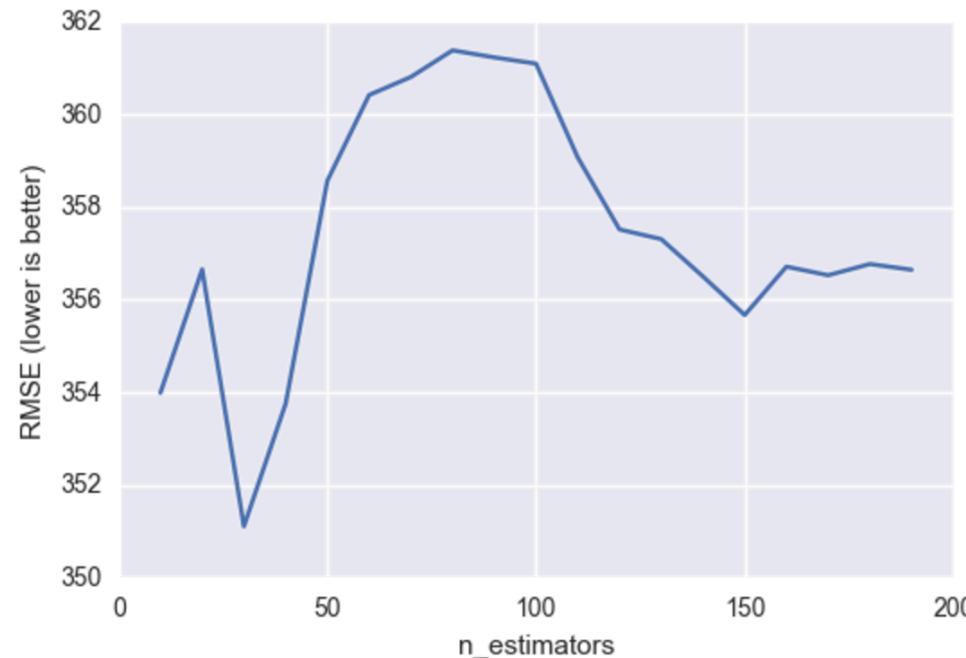


# Results: RandomForestRegressor

My random forest performed worse, RMSE = 313.16

- Again, chat velocity was the only thing that didn't introduce noise

<matplotlib.text.Text at 0x11e6daf10>





# Follow-up: BayesianRidge

Investigated using the chat during the clip to predict plays

- Smaller sample (1,000) clips, produced ~150,000 messages
- Used: Body, Tags

```
In [65]: np.sqrt(metrics.mean_squared_error(df3_comb.total_plays, df3_comb.total_plays_prediction))
```

```
Out[65]: 93.238281403080364
```

```
In [62]: np.sqrt(metrics.mean_squared_error(y_test, preds))
```

```
Out[62]: 11.505129508483178
```

# Next Steps

- Better Feature Engineering
  - More flexible with how clip creation time is mapped to engineered features
- Continuous NLP to see if a running sentiment analysis would be useful for identifying clips
  - Tune NLP and use it in conjunction with the broader model
  - I've hypothesized that emoticons like PogChamp & Kreygasm could indicate highlight worthy moments
    - Internal sentiment scoring may be available
- Features around the person clipping
  - Compare clipper's popularity on the popular social sites
  - Reddit and Twitter API would be the most useful based on my preliminary research



# My Takeaways

- Build a MVP feature before investing more time
  - I spent too much time trying to create great features without testing simplified versions first (subs, bits messages, etc.)
- Expand upon promising features
  - Knowing that chat velocity was important, I could have measured the change in chat velocity over a grouping of seconds or whole minutes
- Samples
  - With a bottom heavy distribution, I found it difficult to find a representative sample and reduce my margin of error
- Accuracy vs. Resources

# Questions!



[www.twitch.tv](http://www.twitch.tv)