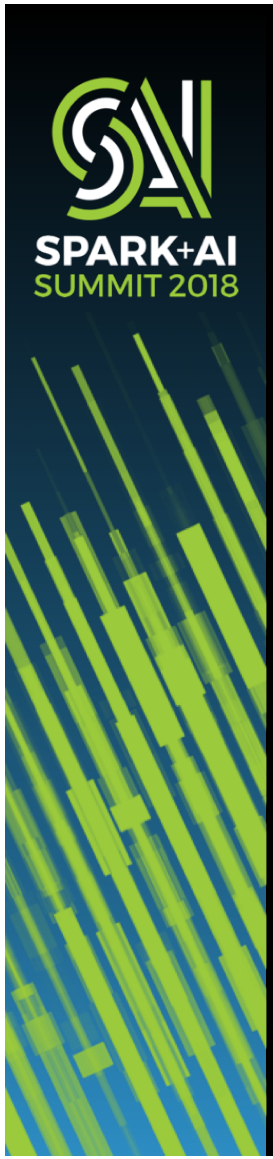


User Predictions from Messy Clickstream Data

Patrick Halina, Zynga

#DSSAIS15

Application Overview

- User installs game
- Capture user actions from first session
- Predict if user will play again in 7 days

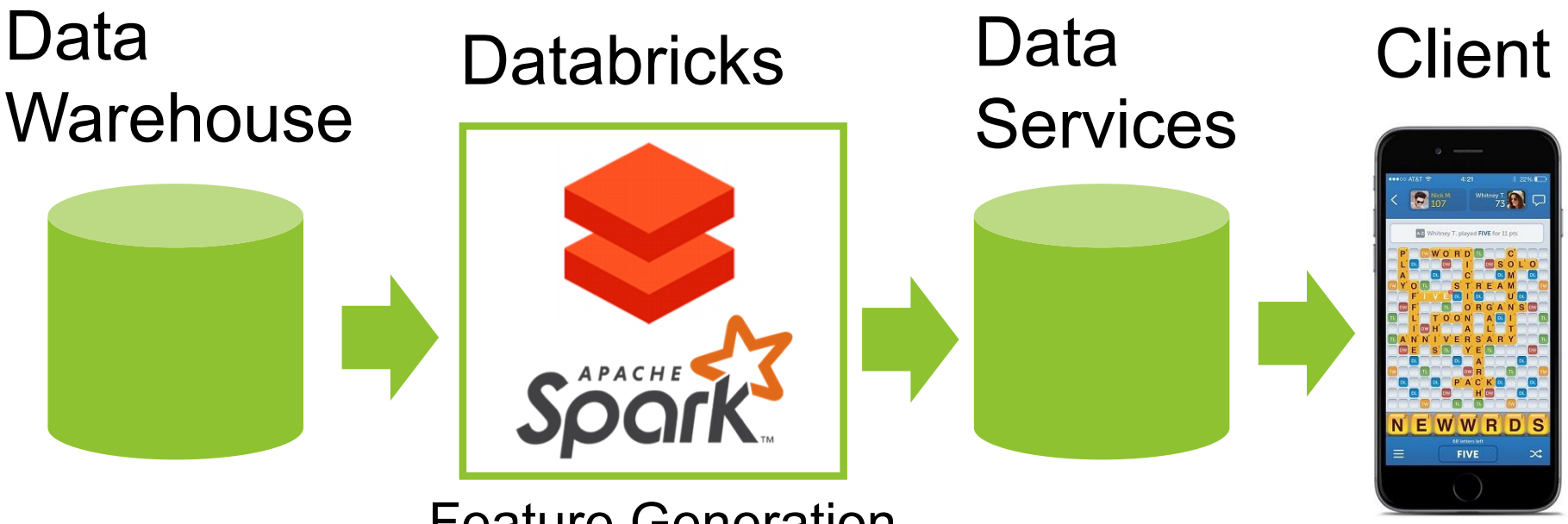
First Session

About Me (Patrick Halina)

- Undergrad in Comp Eng
- Grad school in Statistics
- Previously worked on ML Platform at Amazon
- Tech lead for ML Eng at Zynga, based out of Toronto office

Over 80 million monthly mobile users
Mission: Connect the world through games

Batch ML Predictions at Zynga



ML Challenges

- Building features to train ML models is hard
- At Zynga, feature generation pipeline takes the most time in model dev process
- Garbage in → Garbage out

Why is Feature Generation Hard?

- Need to select important features
- Want to capture relationships between features
- Typically, each feature is explicitly coded in SQL/SparkSQL/Pandas

Clickstream Data

- Log of user actions
- Difficult to wrangle into features

```
08/09/2018 12:01:18
user=123
type=game_action
subtype=level_up
level=12
```

```
10/09/2018 15:29:01
user=124
type=purchase
price=12.99
```

Problem 1: Messy

- Huge logs, typically largest dataset
- Thousands of event types
- Different structure/interpretation for each event type
- Event catalog erroneous and incomplete

Problem 2: Wrong Data Shape

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^m \\ 1 & x_2 & x_2^2 & \dots & x_2^m \\ 1 & x_3 & x_3^2 & \dots & x_3^m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^m \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_m \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \vdots \\ \varepsilon_n \end{bmatrix},$$

User Feature Matrix

Problem 2: Wrong Data Shape

- Most ML models need input as a matrix
- Every matrix input to model needs to have the same features
- Challenge: transform sequence into matrix

Problem 3: Temporal Info

- How to capture timing info of events?
- Eg. Increasing/decreasing trends
- Eg. User plays multiplayer game before trying single player mode

User	*A*	*B*	*C*
Num multiplayer battles	*5*	*0*	*23*
Num single player battles	*22*	*12*	*3*

Traditional Solution

- Select events, aggregate over time period

User	*A*	*B*	*C*
Total Num Battles	*22*	*6*	*99*
Num Battles Last 7 Day	*10*	*6*	*0*
Max Level	*5*	*2*	*17*

Traditional Solution

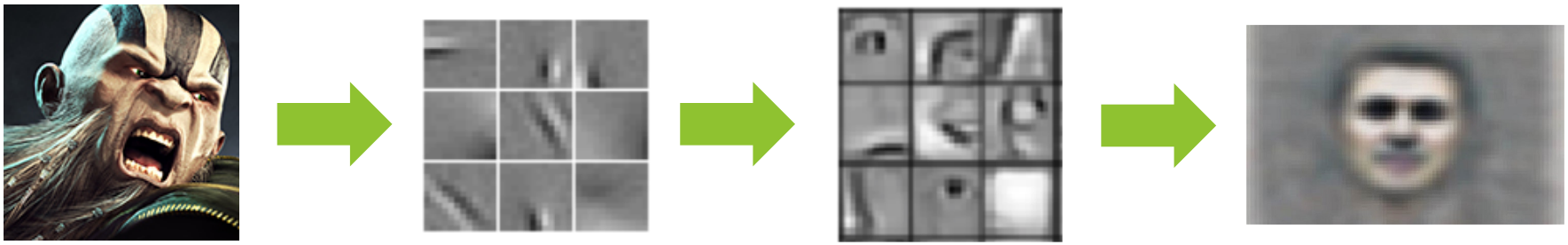
- Human interpretation of events
- Explicitly code features for each event
- Add few temporal signals to code as features
- Eg. Week over week change in battles

Problems

- Takes too much time
- Repetitive
- Hard to debug, maintain
- Miss signals by hand picking features to add

Deep Learning?

- Trend with Deep Learning is to let algorithms select high level features from data
- Deep Learning outperforms handmade features



Deep Learning?

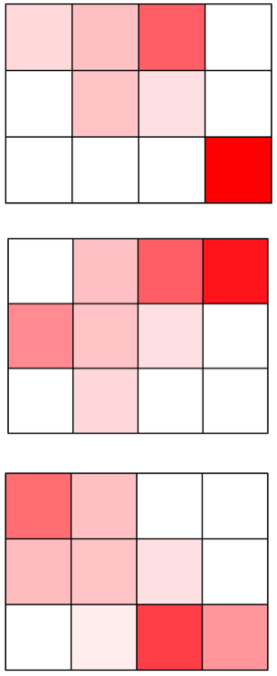
- How can we apply this to event sequences?
- Theoretically: Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM)
- Area of active research, tricky to apply in practice

Solution: Temporal User Heatmap

```
08/09/2018 12:01:18
user=A subtype=level_up
level=12

08/09/2018 12:02:11
user=B subtype=purchase
amount=12
```

Clickstream

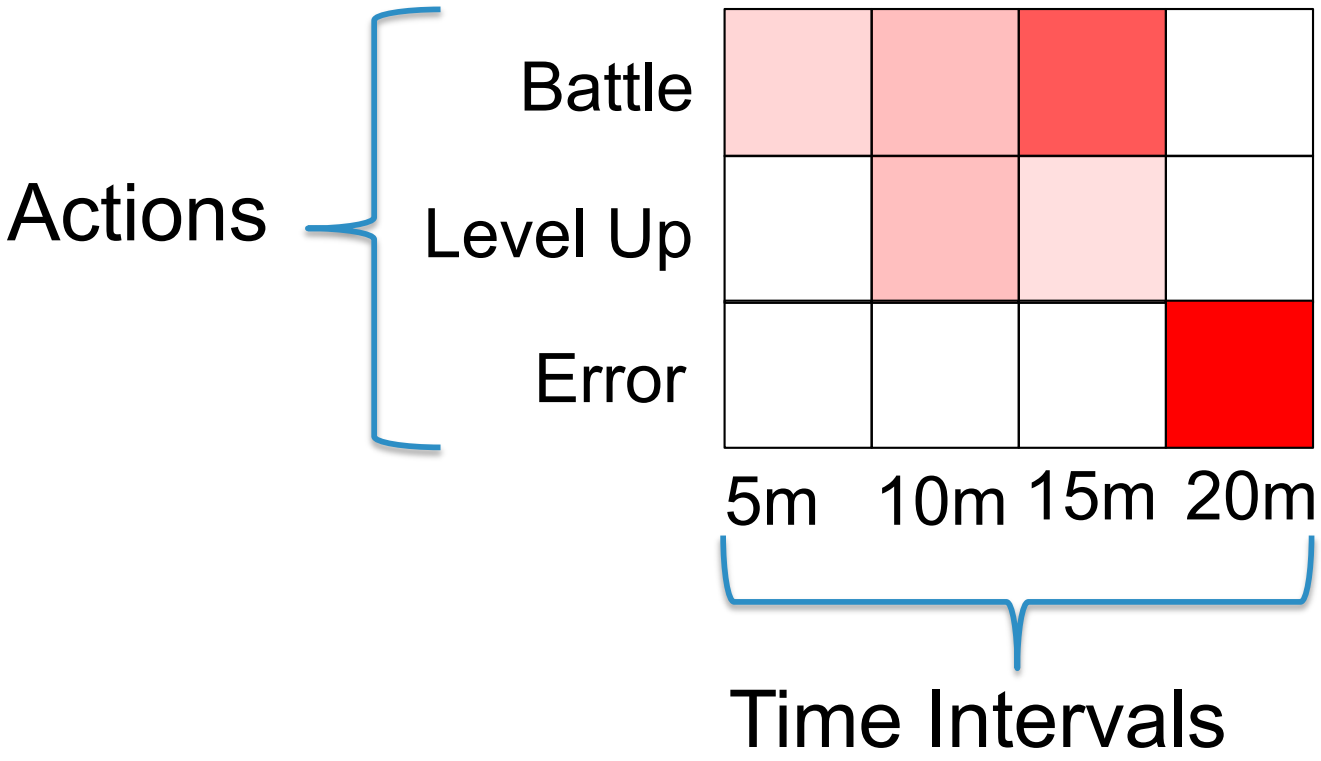


User A

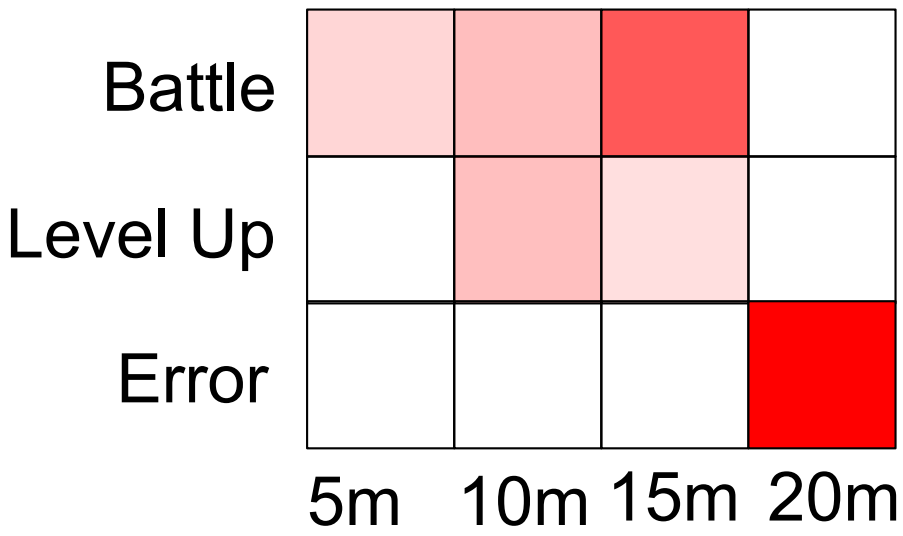
User B

User C

Solution: Temporal User Heatmap



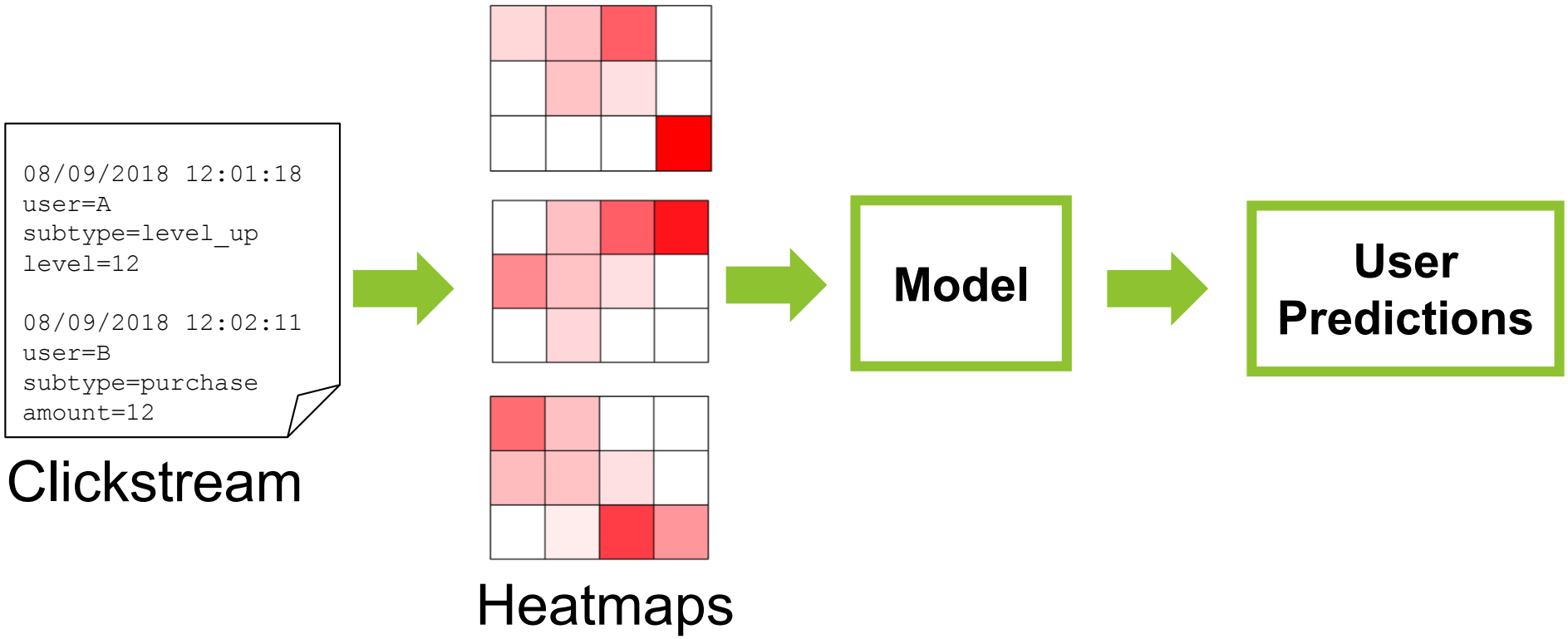
Solution: Temporal User Heatmap



User increased battles over first 15 minutes, then stopped

User had error in last 5 minutes

Dataflow

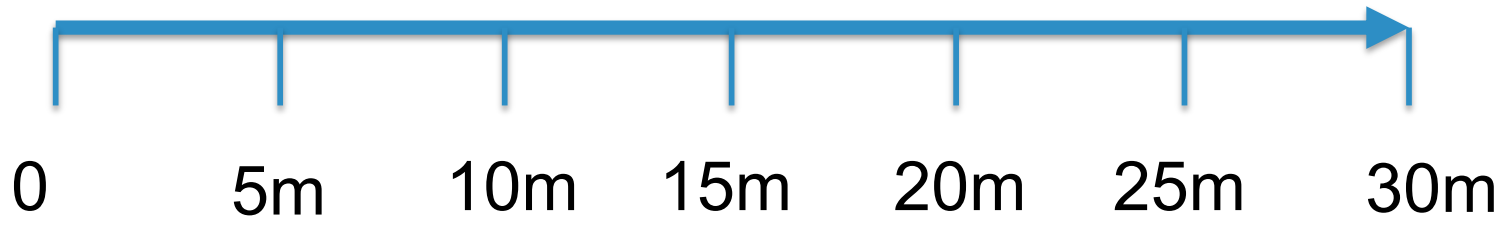

08/09/2018 12:01:18
user=A
subtype=level_up
level=12
08/09/2018 12:02:11
user=B
subtype=purchase
amount=12
Clickstream
Heatmaps
Model
User Predictions

Advantages

- Heatmap is auto generated
- No manual interpretation of events
- Right shape for ML models
- Captures temporal relations
- Simpler than advanced sequence learning models like RNNs and LSTMs

Methodology 1: Timing

- Choose total time window
- Choose time intervals to break up window
- We applied this to new installers: 5 minute intervals during 30 minutes after install



Methodology 2: Calculate Intensities

- Aggregate actions over each time period
- Choose aggregation functions
- Simple aggregation: count occurrences
- Other aggregation functions (max, sum) require interpretation of events but give more info
- Normalize and scale values in heatmaps

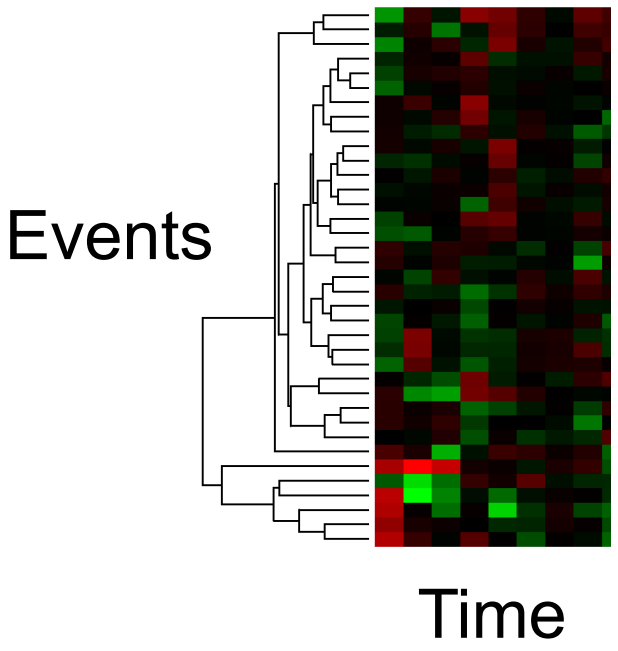
Methodology 3: Limit Events

- Roll up events to limited hierarchy
  `purchase | car | Audi >`
  `purchase | car`
- Select top events by significance to predictor, presence in population
- ie. Decision Tree metrics (Gini impurity, Information gain)

Implementation

- Use Spark
- Developed generic Python framework at Zynga to generate features from any game's events
- No need to write custom feature generation for each game

Modeling



- Rearrange heatmap so similar events are closer to each other
- Use hierarchical clustering
- Now grid cells are related to nearby cells by both time and correlation

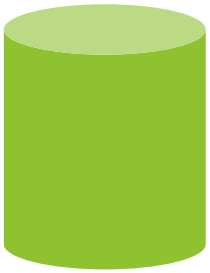
Modeling

- Clickstream events are now an image
- Apply models that capture spatial structure between columns and rows of data
- Eg. Image classification techniques

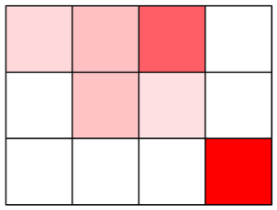
Retention Prediction

- Capture user game actions from first 30 minutes after installation
- How well can we predict retention?

First Session



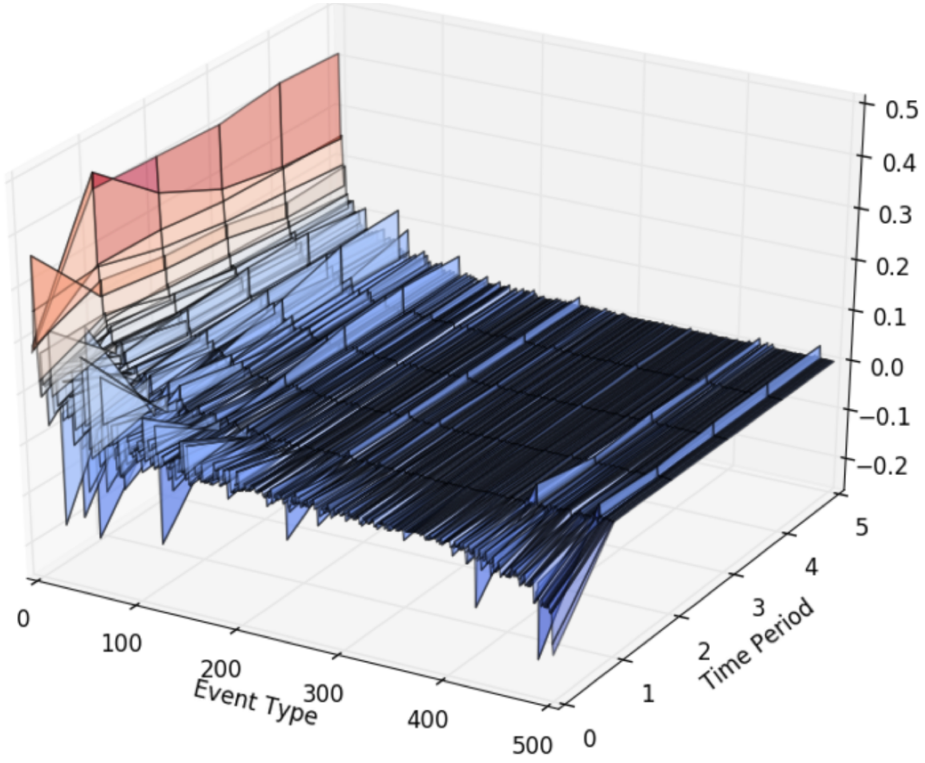
Clickstream



Heatmap

User Predictions

Retention Prediction

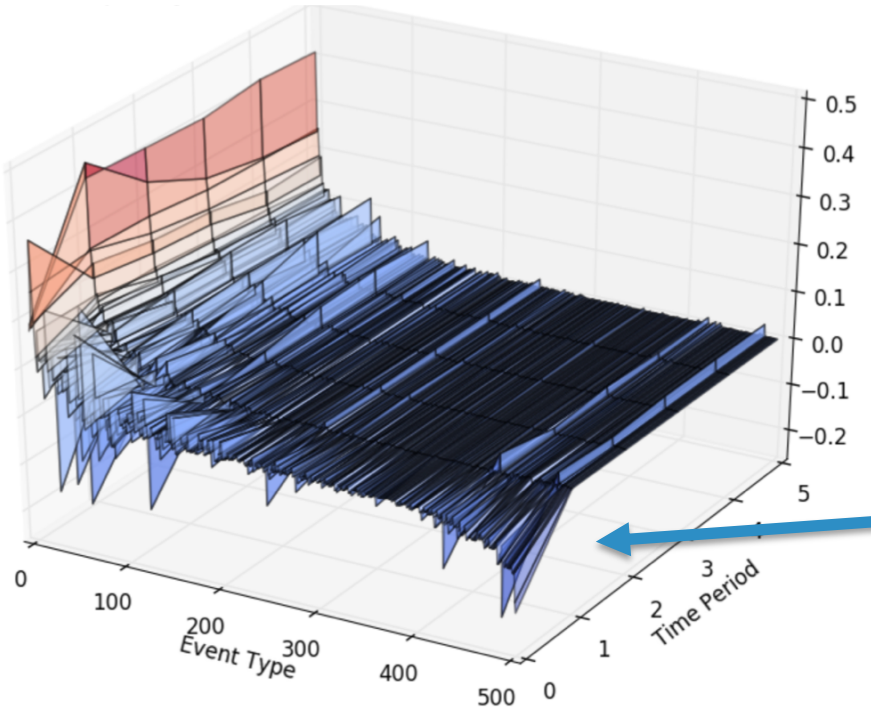


- Looked at new installers for first 30 minutes of gameplay
- 500 events, 5 minute intervals
- Compared users who retained vs. quit

Retention Prediction

Difference between installers who stayed vs. quit

Big difference in some actions between users who stay active vs. become inactive

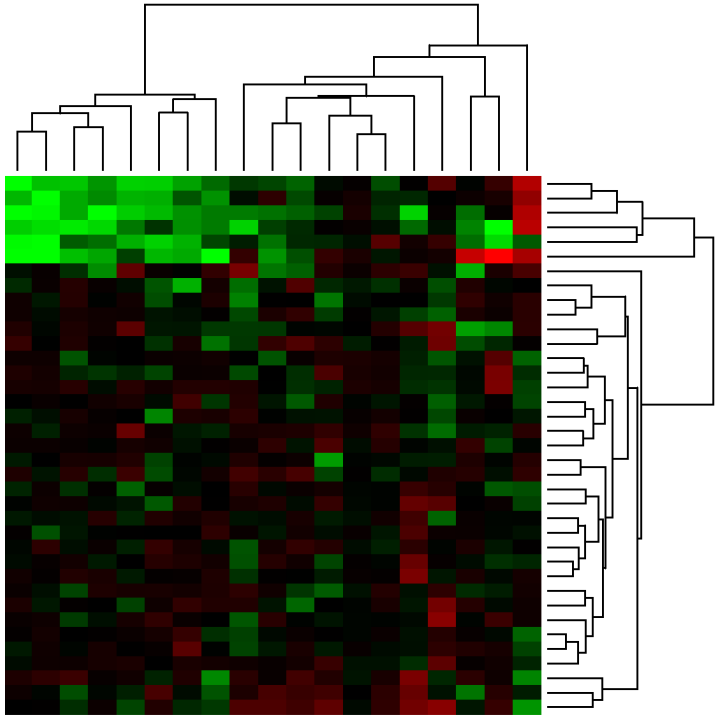


A few events more likely to happen for quitters (eg. Error)

Modelling Results

- Logistic regression is poor, doesn’t capture spatial relationships
- Random Forest performed very well
- Deep Neural Network needs custom network, work in progress

Other Applications



- Make predictions on longer time frames
- Event significance to provide insight on why users stop playing
- User clustering

Thank You!

- Special thanks to Mehdi Ben Ayed and Jijun Xiao