Wargaming: Analyzing WoT gamers’ behavior

Nishith Pathak
n_pathak@wargaming.net
Wargaming BI
14/03/2018

2

WARGAMING.NET
LET'S BATTLE

Wargaming

WARGAMING.NET
LET'S BATTLE

And more!

WARGAMING.NET
LET'S BATTLE

Wargaming in the Czech Republic

Wargaming Czech Republic

Newest addition to the Wargaming Empire

Wargaming Prague

QA, Data Warehouse, Distributed Development, Back office support, Global Procurement and Legal services

Wargaming Brno

Prototyping and development of World of Tanks gameplay

Friendly and fun environment

Collaborate on cutting edge projects with colleagues from around the globe

Exceptional opportunities for professional growth

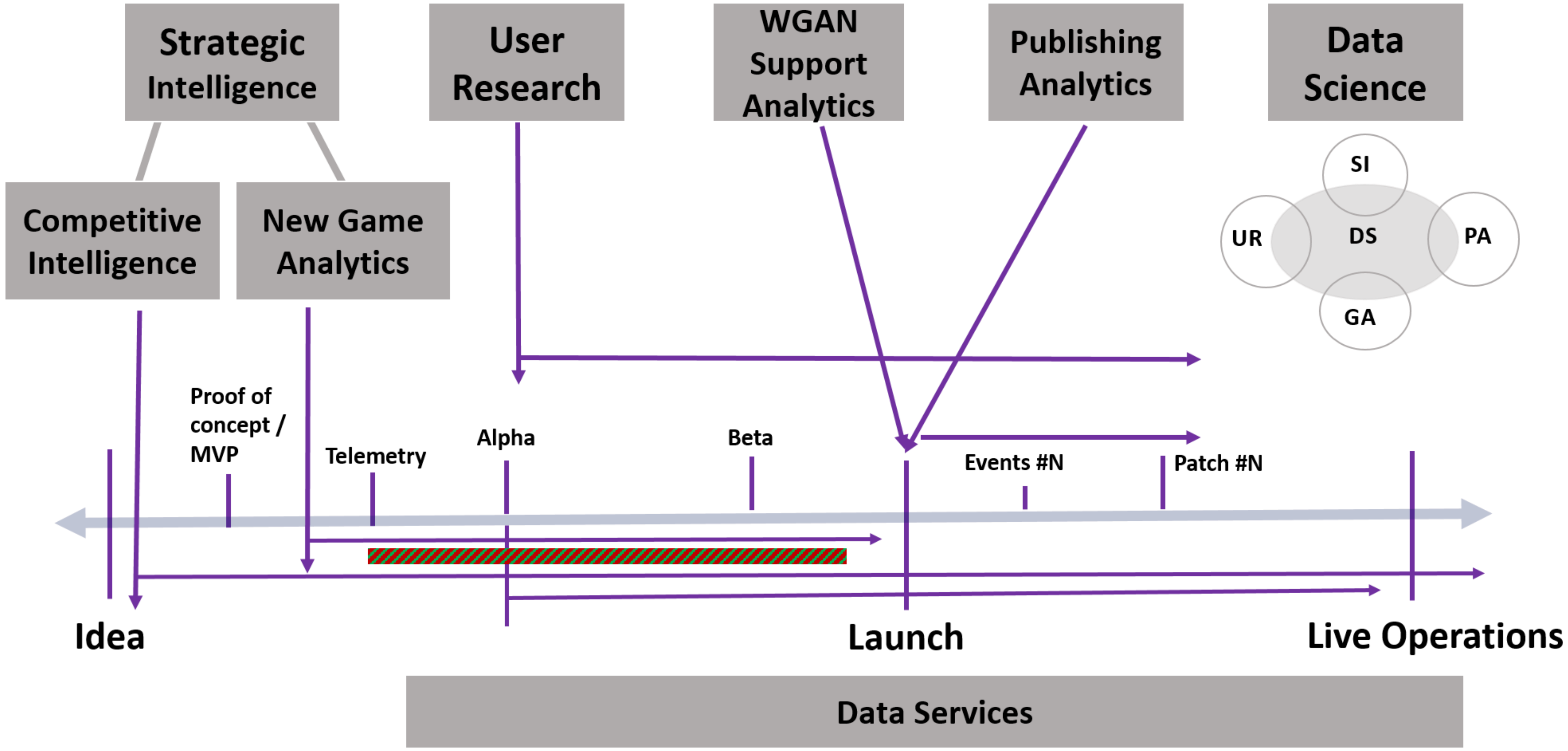
WARGAMING.NET
LET'S BATTLE

BI Locations and Structure

BI Locations	Americas Austin	HQ Nicosia	CIS Minsk	APAC Singapore	Europe Prague
Analytics Services	Strategic Intelligence	Support Analytics	User Research	Publishing Analytics	Data Science
Data Services	Data Services Architecture		Data Services Operations		Data Services Operational Leadership

5

BI in the product lifecycle



WARGAMING.NET
LET'S BATTLE

Core focus of Data Science

Develop models and algorithms in support of all BI functions

Support regional publishing and game analysts with complex product analyses

Support Player Relationship Management Globally

Explore new technologies, methodologies, and develop new tools

WARGAMING.NET
LET'S BATTLE

How data science supports each BI team

Strategic Intelligence

- Telemetry Input
- Feature Analysis
- Life Cycle analysis

User Research

- Player Satisfaction
- User Profiling
- PRMP Surveys

Game Support

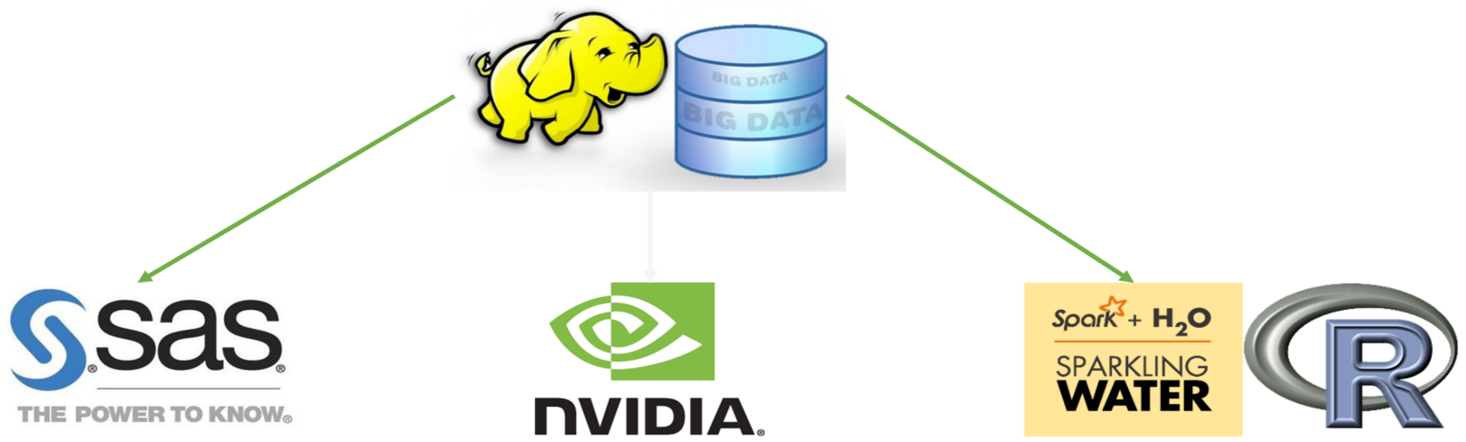
- Cheat/Bot Detection
- User Segmentation
- Progression Models

Publishing Support

- CRM Support
- CS Models
- LTV Models

Data science tools

Data Warehouse



- Engineered Solution
- Massive Parallelization
- Model Management
- Limited Algorithms

- GPU Processing
- Training Parallelization
- No model management
- Cutting-edge Algorithms

- Open Source Solution
- In-depth Customization
- Limited Model Management
- Cutting-edge Algorithms

WARGAMING.NET
LET'S BATTLE

Analyzing player behavior in non-contractual settings

- *How many games is each user expected to play in the next 90 days/3 months?*
- Translate RFM measures to the gaming domain –
  - *Recency:* When was the last time user A played the game?
  - *Frequency:* How often does user A play the game?
  - *Monetary / Intensity:* Every time user A has a game play session, how many games on average does he/she play?

Analyzing player behavior in non-contractual settings

WARGAMING.NET
LET'S BATTLE

- The "Buy till you die" (BTYD) family of models
  - Probabilistic models for user behavior in non-contractual settings
  - Developed by the marketing research community
  - Common theme: *Recurrent survival model which allows users to churn from the process*

David C. Schmittelein, Donald G. Morrison, and Richard Colombo. 1987. Counting Your Customers: Who Are they and What Will they Do Next? Management Science 33, 1 (1987), 1–24. DOI:http://dx.doi.org/10.1287/mnsc.33.1.1

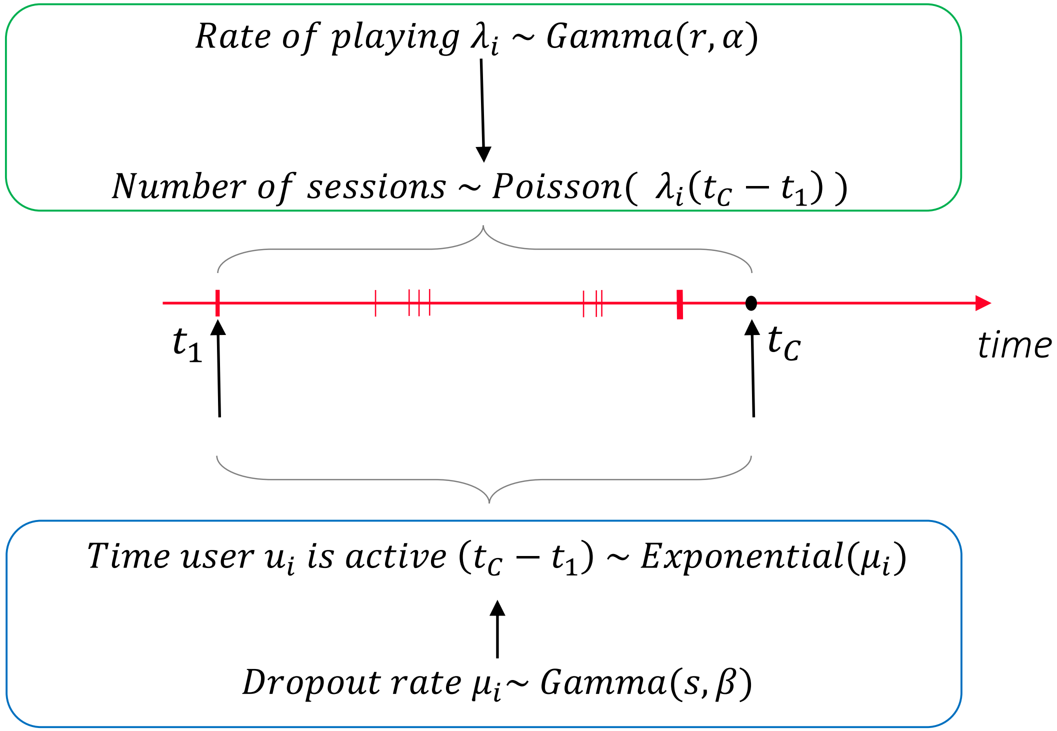
Peter S. Fader, Bruce G. S. Hardie, and Ka Lok Lee. 2005. Counting Your Customers? the Easy Way: An Alternative to the Pareto/NBD Model. Marketing Science 24, 2 (2005), 275–284. DOI:http://dx.doi.org/10.1287/mksc.1040.0098

Peter S. Fader, Bruce G. S. Hardie, and Ka Lok Lee. 2005. RFM and CLV: Using Iso-Value Curves for Customer Base Analysis. Journal of Marketing Research XLII, November (2005), 415–430. DOI:http://dx.doi.org/10.1509/jmkr.2005.42.4.415

WARGAMING.NET
LET'S BATTLE

The "Buy Till You Die" family of models

- Model how long a user is active
  - Dropout rate parameter μ_i
- Model how many sessions and games the user plays *while* he/she is active
  - Playing rate parameter λ_i
- Many different variations based upon different choices of modeling playing and dropout behavior



WARGAMING.NET
LET'S BATTLE

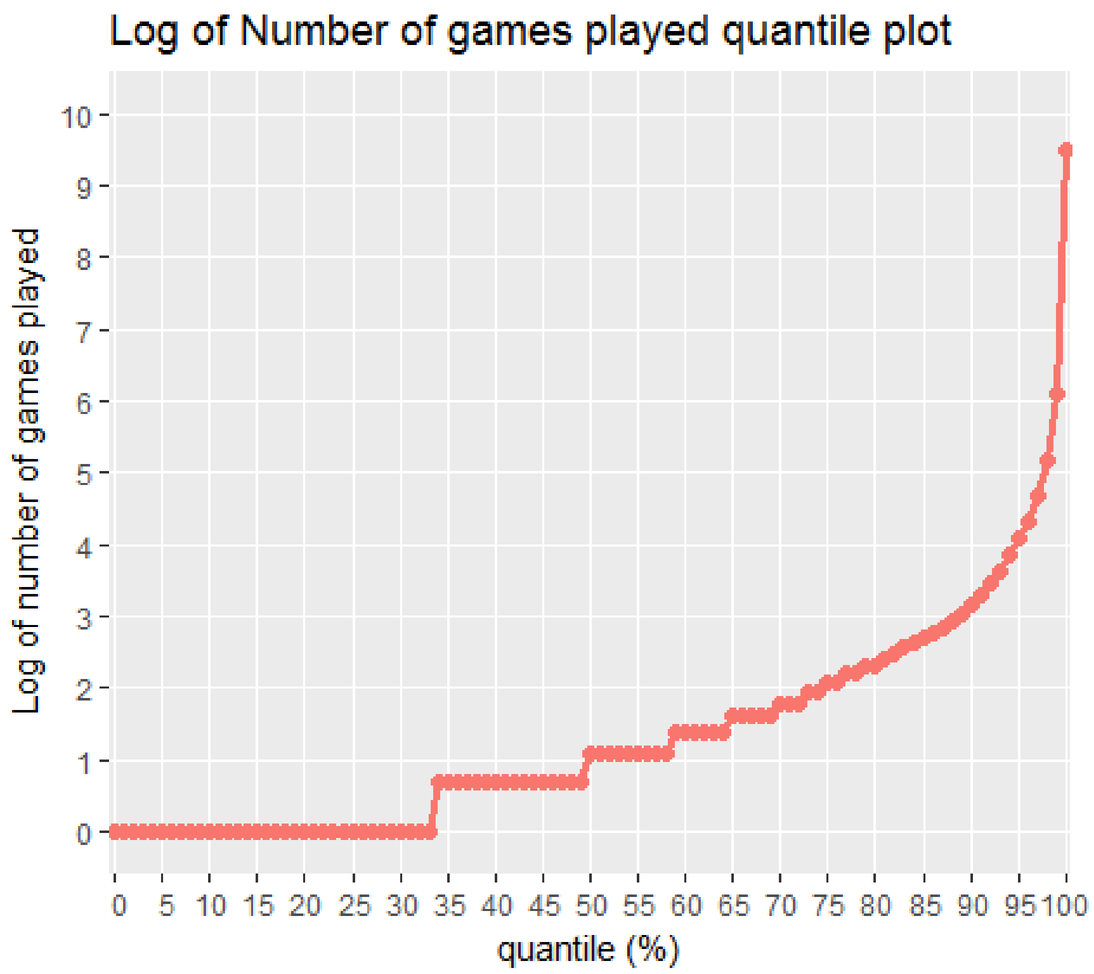
The “Buy Till You Die” family of models

- Works with just activity log data
  - Privacy concerns, users don’t want to share personal information, users misreport information, registration process should be less intrusive.
- Parameters determined using maximum likelihood estimation
- Useful for summarizing and predicting population level trends

13

Predicting user behavior with BTYD

- Dataset consists of all EU region players joining between 1st Feb 2016 and 1st May 2016
- All users' games ,for each day, are tracked till 1st May 2017
  - Number of users = 331,811
  - Number of games = 206,897,542
- Data from 1st Feb 2016 – 31st Jan 2017 was used to predict *number of games* each user will play in the next 90 days



WARGAMING.NET
LET'S BATTLE

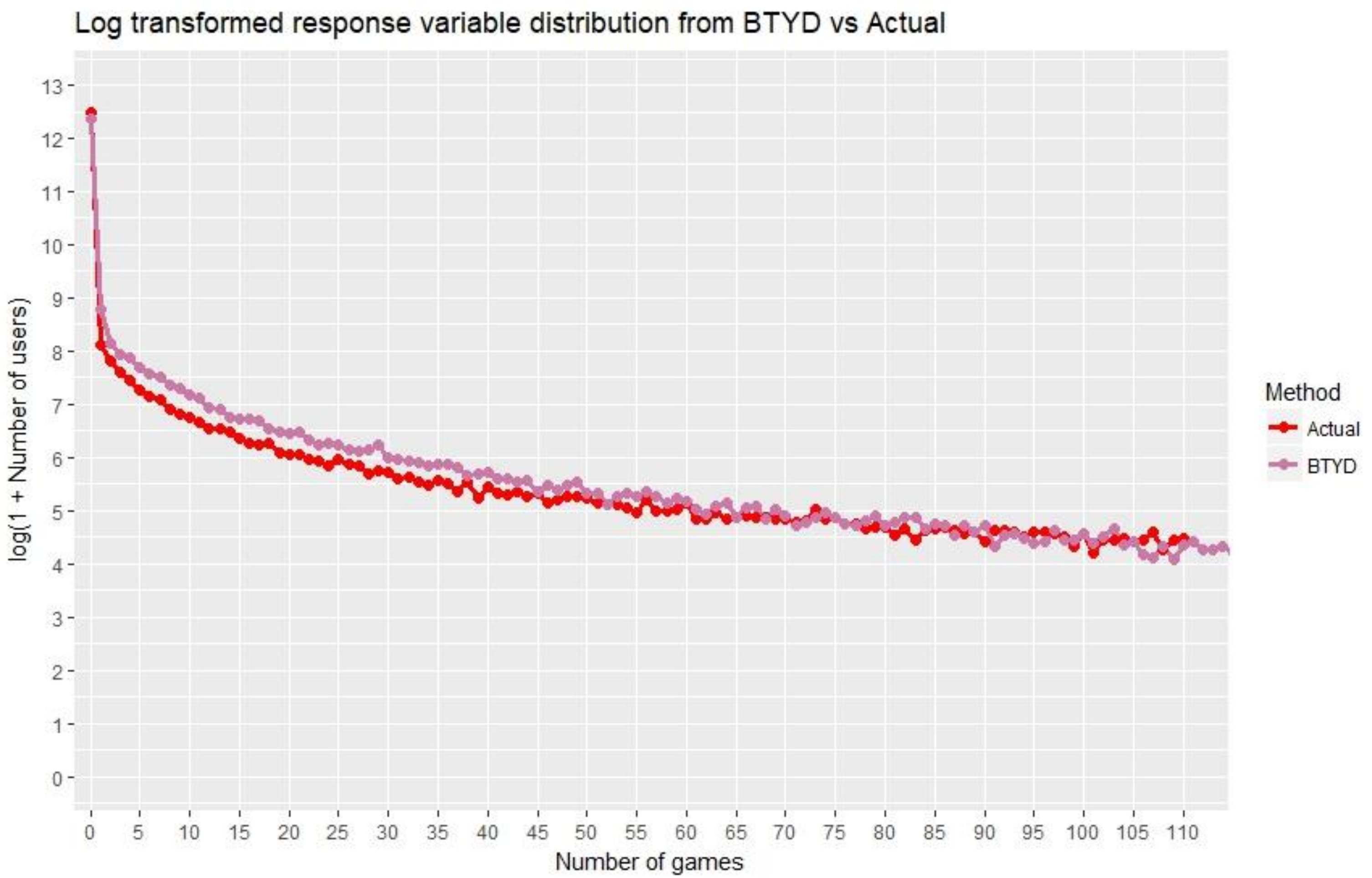
Predicting user behavior with BTYD

- RFM based features prepared for each user
  - *Frequency*: Number of active days
  - *Recency*: Number of days since last login
  - *Intensity*: Number of games played
  - *Tenure*: Number of days between first and last active days

WARGAMING.NET
LET'S BATTLE

Predicting user behavior with BTYD

- Total number of games played predicted by BTYD: *28,852,874*
- Total number of games actually played: *31,329,849*



WARGAMING.NET
LET'S BATTLE

Individual predictions with BTYD

- *Whales*: Few users having a very large number of games
- Exclude top 10% players based on number of games played
  - Total number of users = 300,399 (90%)
  - Total number of games played = 68,834,316 (33.27%)
- *Pareto principle*

WARGAMING.NET
LET'S BATTLE

Error Metrics for evaluating individual predictions

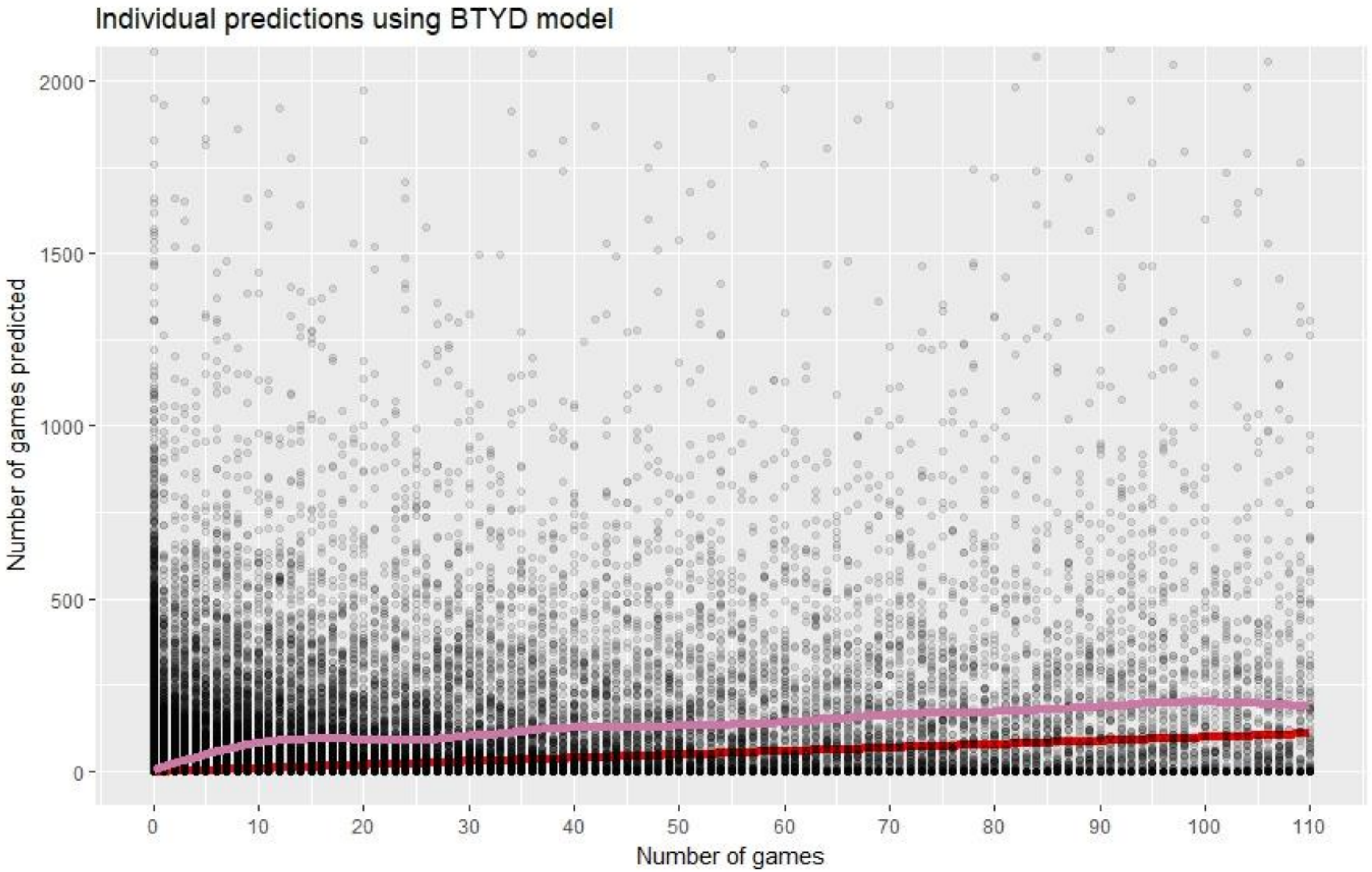
- *Root mean squared error* (RMS Error) = Square root of the mean of squared error
  - Squared error for a given user u_i is $\mathrm{e_i} = (y_i \ - \hat{y}_i)^2$

- *Mean absolute error* (ABS Error) = Mean of absolute error
  - Absolute Error for a given user u_i is $\mathrm{e_i} = |y_i \ - \hat{y}_i|$

- *Mean relative absolute error* (Rel. ABS Error) = Mean of relative absolute error
  - Relative absolute error for a given user u_i is $\mathrm{e_i} = \frac{|y_i - \hat{y}_i|}{y_i}$
  - But $y_i = 0$ is an issue and so we use $\mathrm{e_i} = \frac{|y_i - \hat{y}_i|}{\max(1, y_i)}$

18

Predicting user behavior with BTYD



WARGAMING.NET
LET'S BATTLE

Machine learning for predicting user behavior

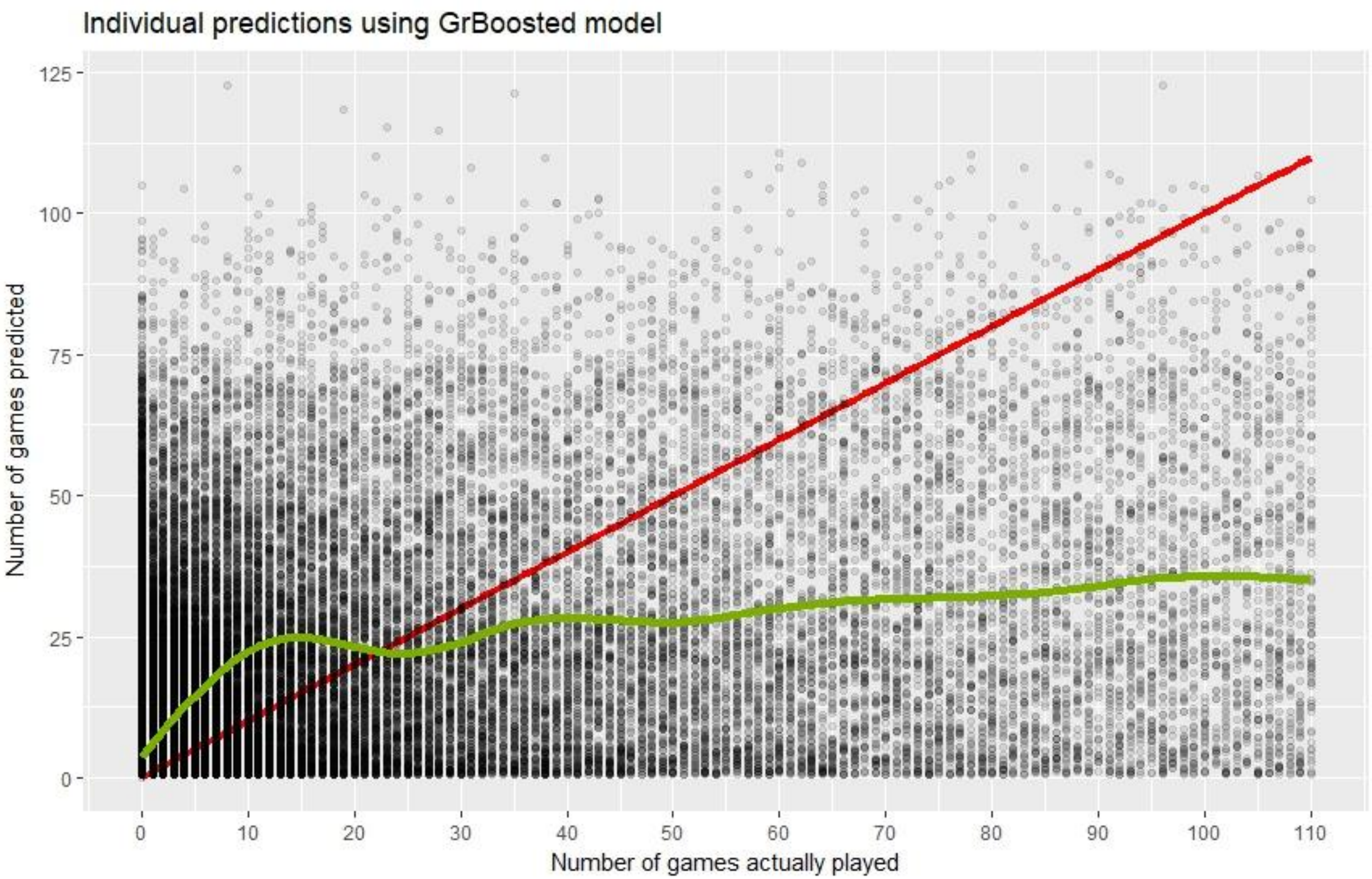
- *Gradient Boosting*
  - State of the art before Deep Networks, still continue to be one of the best machine learning techniques for learning from data
  - Used to win several data science competitions/challenges
- *Core idea*
  - Train a model on data
  - Train another model which learns where the first one makes mistakes
  - Train another model which learns where the combined model makes mistakes
  - Keep repeating the previous step!

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. Annals of Statistics, pages 1189–1232.

Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16). ACM, New York, NY, USA, 785-794. DOI: https://doi.org/10.1145/2939672.2939785

20

Predicting user behavior with Gradient Boosting



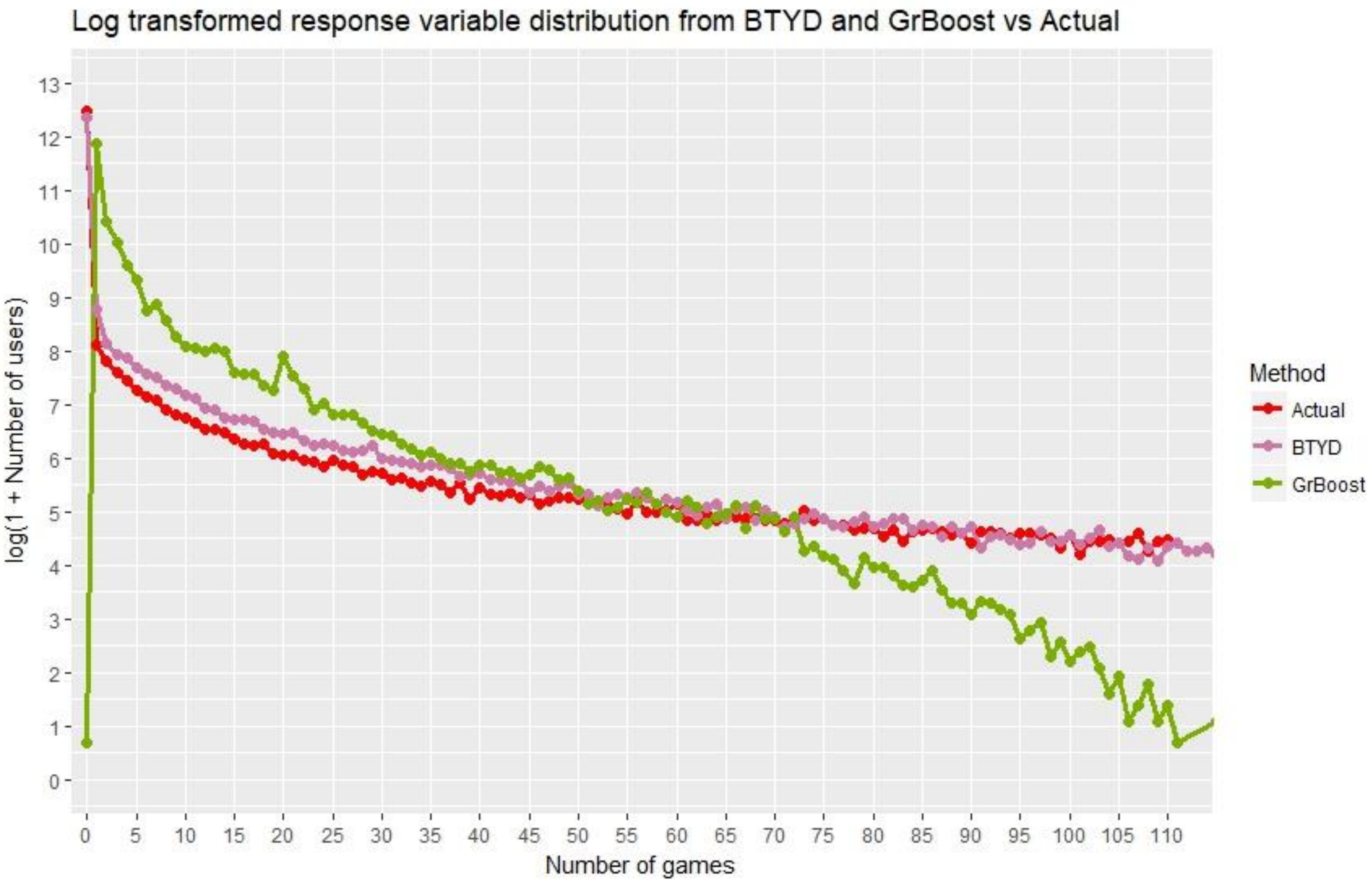
BTYD vs Gradient boosting for predicting number of games played

- Does significantly better than BTYD on the individual predictions but loses some of population level properties captured by BTYD

	BTYD	*GrBoost*
RMS Error	80.57	12.27
Mean ABS Error	15.09	5.78
Mean Rel. ABS Error	5.38	3.52

WARGAMING.NET
LET'S BATTLE

Comparing response variable distributions – Gradient Boosting



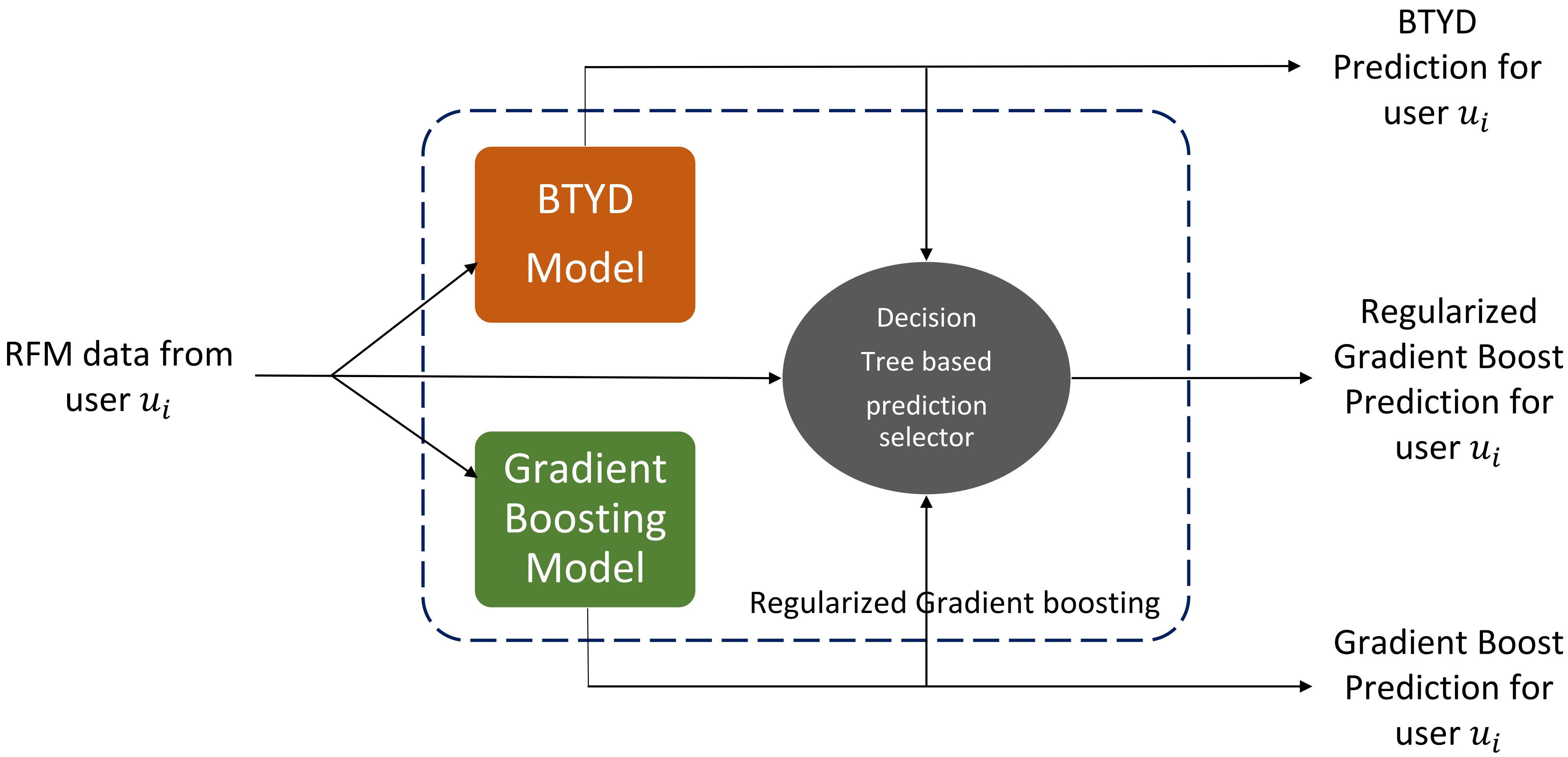
WARGAMING.NET
LET'S BATTLE

Combining Gradient Boosting and BTYD

- Boosting fixes the larger errors in BTYD predictions but does so at the cost of overestimating number of users having fewer games
- BTYD is better at capturing the overall distribution of the response variable but does poorly on individual estimations
- *Can we have the best of both worlds?*

WARGAMING.NET
LET'S BATTLE

Ensemble of Gradient Boosting and BTYD



Combining Gradient Boosting and BTYD

- Learn a decision tree on when to use BTYD predictions and when to use boosting predictions!
  - *RegGrBoost*: Regularizing gradient boosted predictions with BTYD predictions
- For each user use decision tree model to decide whether to use BTYD prediction or Gradient Boosted prediction!

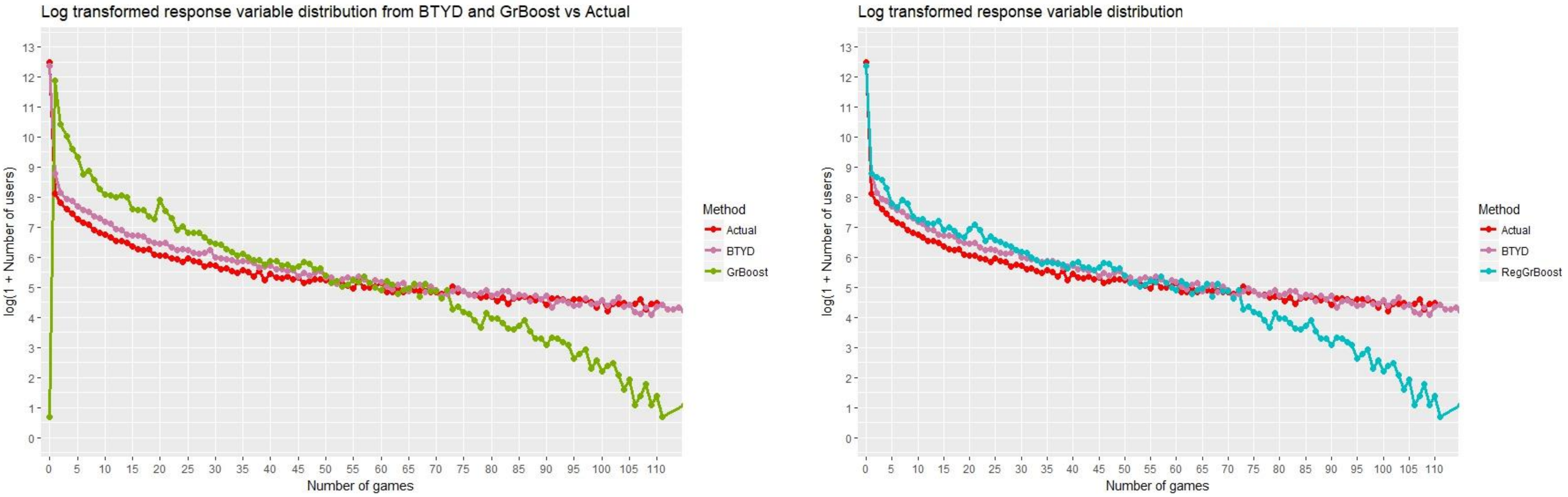
WARGAMING.NET
LET'S BATTLE

Regularized Gradient boosting for predicting number of games played

- Improved predictions on users having fewer number of games played

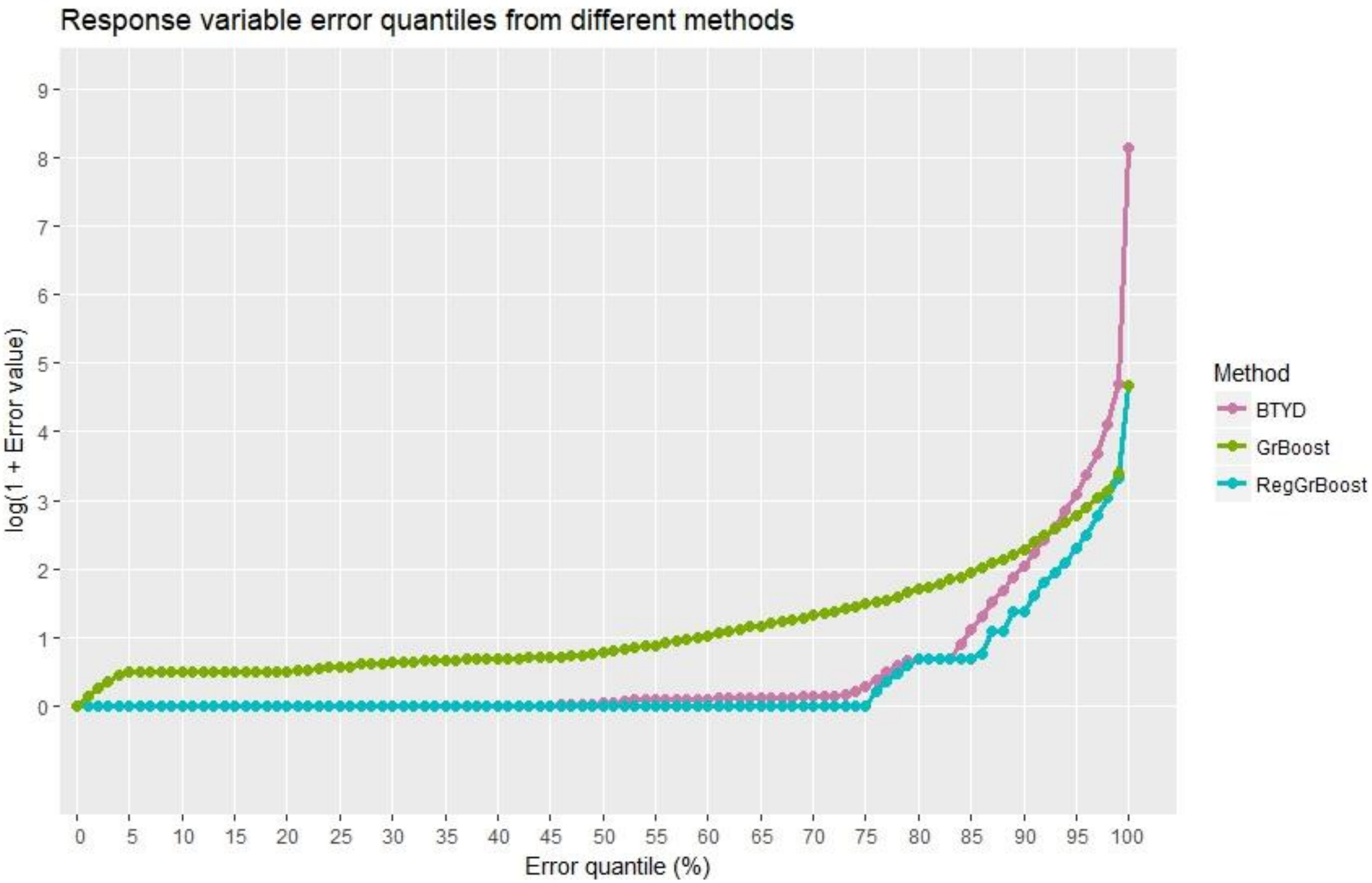
	BTYD	*GrBoost*	*Reg-GrBoost*
RMS Error	80.57	12.27	12.13 (98.86%)
Mean ABS Error	15.09	5.78	3.91 (67.65%)
Mean Rel. ABS Error	5.38	3.52	1.47 (41.76%)

27 Comparing response variable distributions – GrBoost vs RegGrBoost



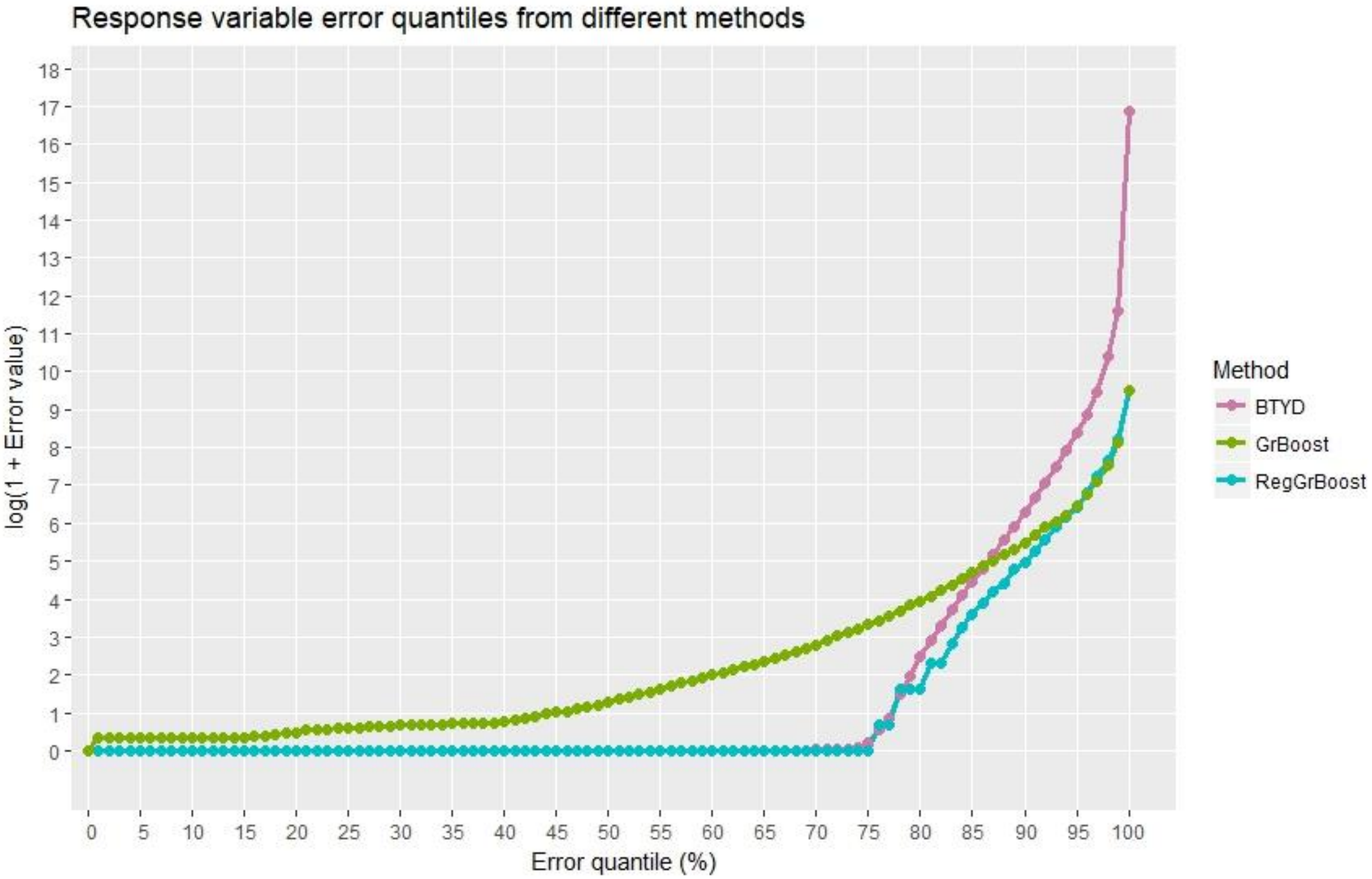
WARGAMING.NET
LET'S BATTLE

Log Rel. ABS error quantile plot for all methods



WARGAMING.NET
LET'S BATTLE

Log squared error quantile plot for all methods



Conclusions and Future directions

- BTYD and Gradient boosting combined to produce a model improving on both
  - Decision tree model used to learn whether to use the response from gradient boosting or the average user behavior predicted by BTYD
  - Can it be applied to other scenarios?
  - Explore from a Machine Learning theory perspective
- Largest errors due to *Winback* phenomenon

WARGAMING.NET
LET'S BATTLE

Conclusions and Future directions

- Plan to use this in support of our new mobile products
- Work with publishing and CRM teams for ROI based evaluation
- Other successful models developed in the past which have been empirically verified to provide ROI lifts for WoT, WoWS and other games

WARGAMING.NET
LET'S BATTLE

THANK YOU!

- *www.wargaming.com*
- *Questions/Answers*

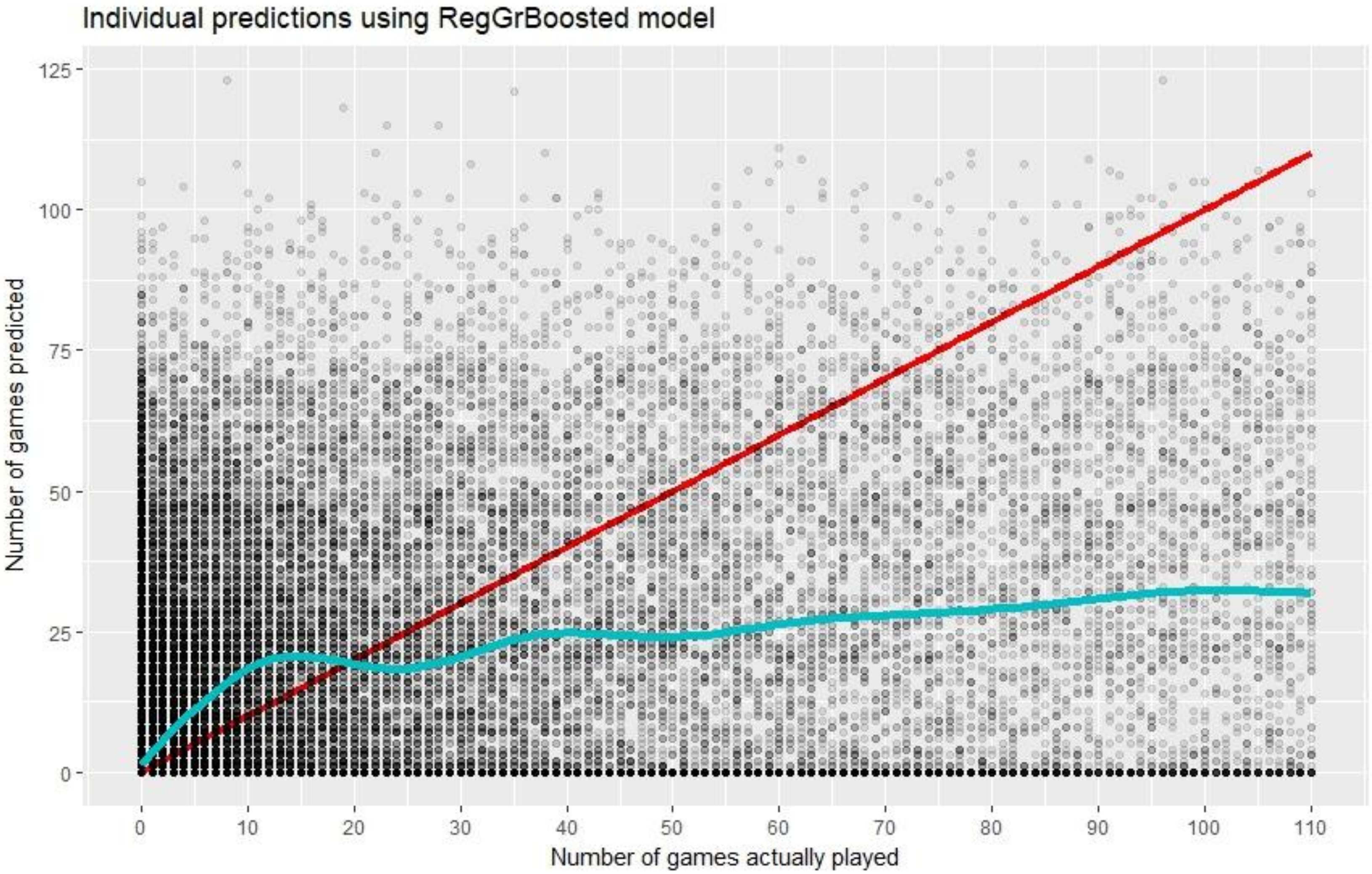
Appendix I – Details on the "Buy Till You Die" family of models

- *Pareto/NBD Model*
  - While customer is active, number of transactions in time $t \sim Poisson(\lambda_i t)$
  - Transaction rate for customers $\lambda_i \sim Gamma(r, \alpha)$
  - Each customer has an unobserved lifetime $\tau_i \sim Exponential(\mu_i)$
  - Dropout rate for customers $\mu_i \sim Gamma(s, \beta)$
  - Transaction and dropout rates vary independently across customers
- *Gamma-Gamma* spending model to estimate expected spend per transaction
- Many different variations based on different choices of modeling transaction and dropout behavior

Appendix II - Gradient boosting for predicting number of games played

- Training data from 1st Feb 2016 – 31st Jan 2017 further split into two parts-
  - ML Training data from 1st Feb 2016 – 1st Nov 2016
  - ML validation data from 2nd Nov 2016 – 31st Jan 2017
- For each user
  - Prepare RFM based features from ML Training data
  - Use corresponding number of games played in ML validation data as response
- Use to train gradient boosted regression tree model
  - Measure prediction errors on test period 1st Feb 2017 – 1st May 2017

Appendix III - Regularized Gradient Boosting predictions



Appendix IV – Decision Tree for choosing prediction model (0: BTYD, 1: Gradient Boost)