playhaven

Churn Prediction With Graphical Models

The goal is to predict if a currently active user will no longer be active next week, within mobile games.

Twitter: @GrimmScientist
Email1: TheGrimmScientist@gmail.com
Email2: allen@playhaven.com
Phone: 724-312-4157

Presenter: Allen Grimm
From: PlayHaven

On: Nov 9, 2013
At: PyData, NYC

Structure of Presentation

- Overview of PlayHaven
- Definition of Churn
- Our Graphical Model Framework
- Experiment 1 – Predictions for one game.
- Experiment 2 – Predictions across the full network.

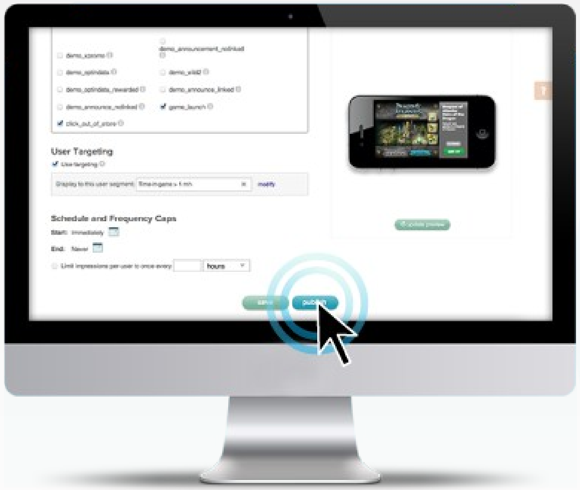
PlayHaven:

- What we do
- How data science can help

Introducing: PlayHaven

The **Business Engine** to help mobile game developers acquire, engage, and monetize players.

SDK



Dashboard

Action

Opportunities for Data Science at PH

- Automate everything
- Response modeling
- Predictive segment dimensions
- Advanced analytics

Player Churn:

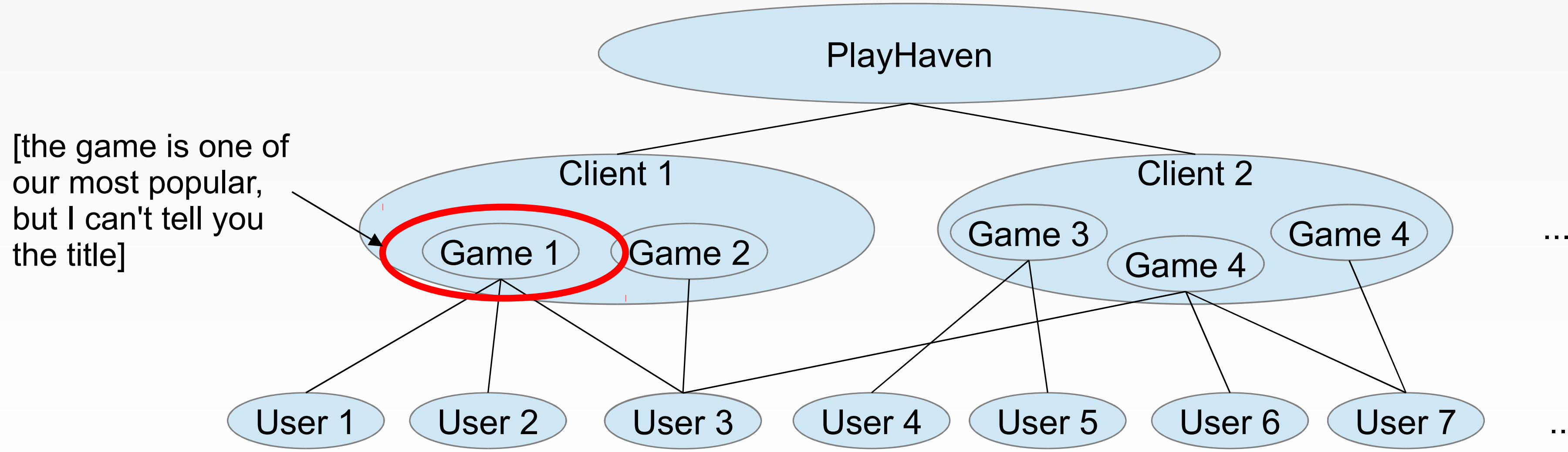
- Let's make a precise definition

Introducing: Player Churn

- There is a precise cost of acquisition for each user (it's never free).
- Can we detect when a user is about to leave?
  - (if so, we can presumably do something about it)
- Churn: When a device used to be active and is no longer active.

Let's start with just one game

- We have lots of publishers with lots of games.
- Let's just pick one game's worth of data to start.



Our Data: Aggregated Game Opens

- We have lots of event types, but let's start simple.

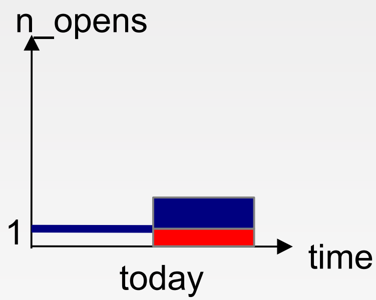
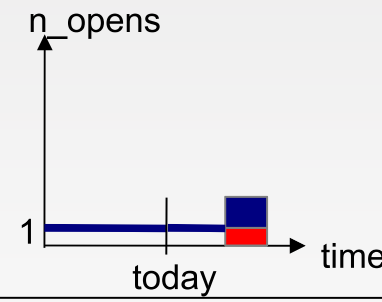
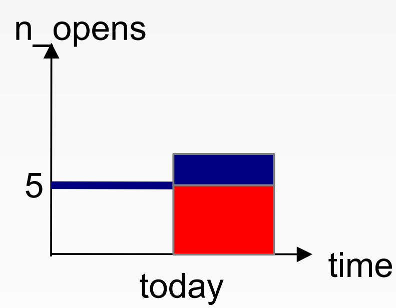
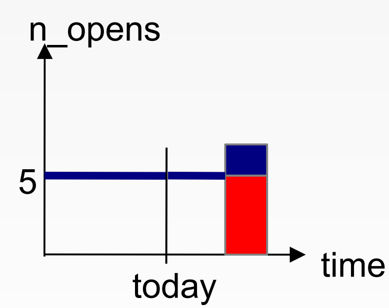
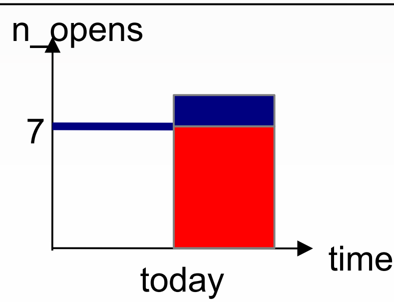
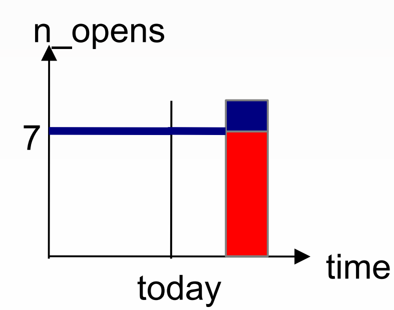
device_id	timestamp
1337	'2012-07-02 00:00:03'
1337	'2012-07-02 14:13:12'
1337	'2012-07-06 11:43:26'
1337	'2012-07-07 11:11:11'
	...

[aggregated opens are what we use internally to calculate a person's activity level, anyway]

- Starts as one case per open.
- Converted to one case per device

device_id = 1337	0	2	0	0	0	1	1	...							

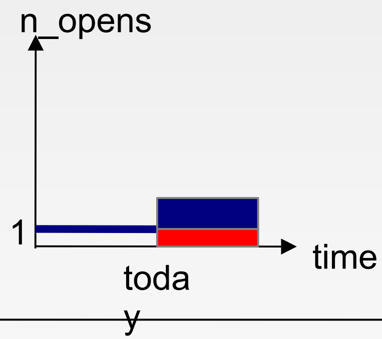
Churn – a user who is no longer active

	Will Churn	Will Churn Soon
Any activity		
Medium activity		
High activity		

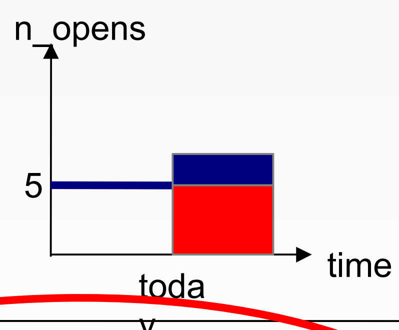
Churn – a user who is no longer active

Just Churned

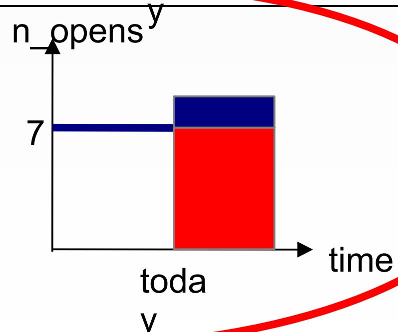
Any activity



Medium-active

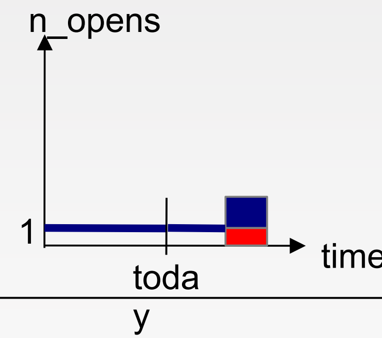


Extremely active

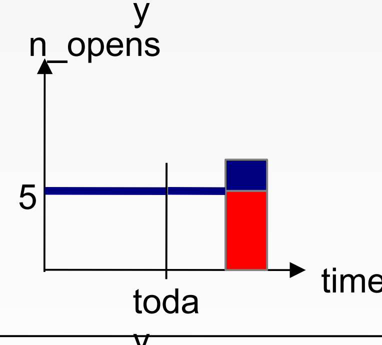


Will Churn Soon

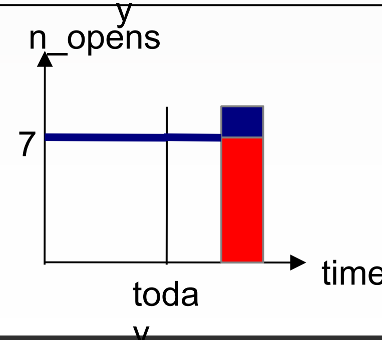
Any activity



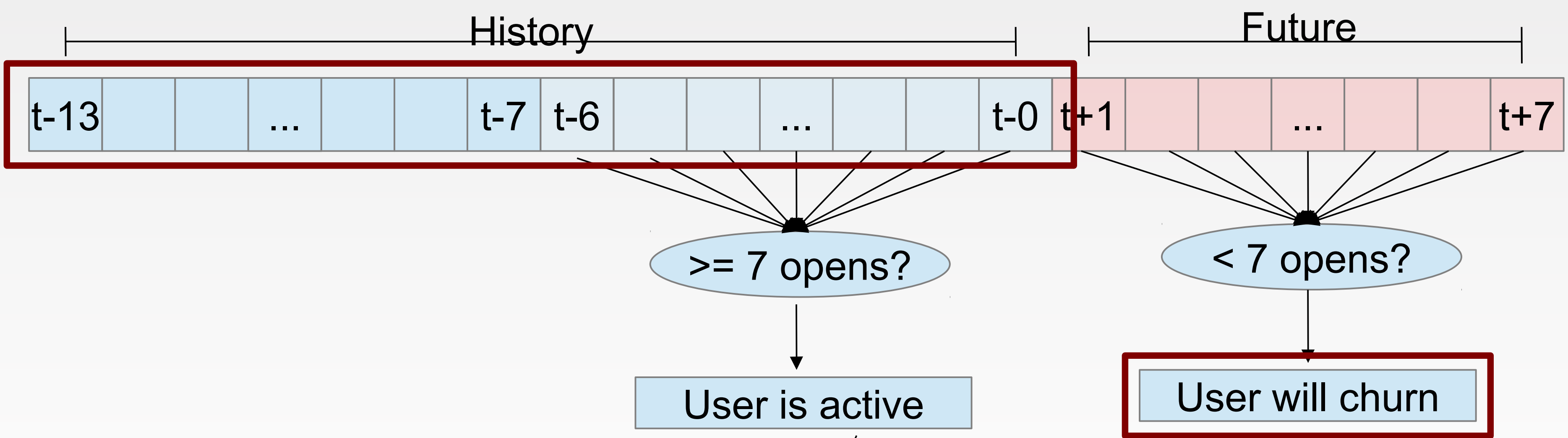
Medium-active



Extremely active



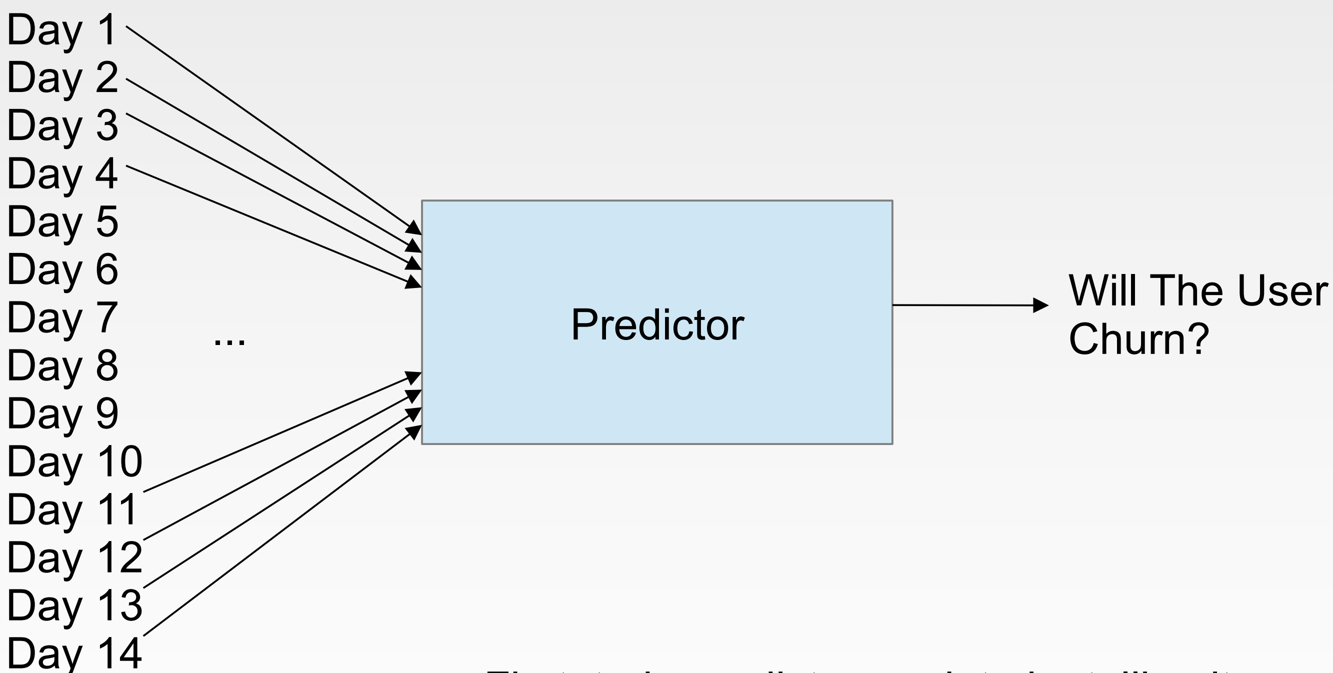
Churn – A Single Sample of Data



-variables used in model are blocked in red.

Case only valid if user is currently active

Our Black Box for Supervised Learning



- First, train predictor on data by telling it both the input and output.
- Second, test the predictor by only giving it inputs and checking what output it gives.

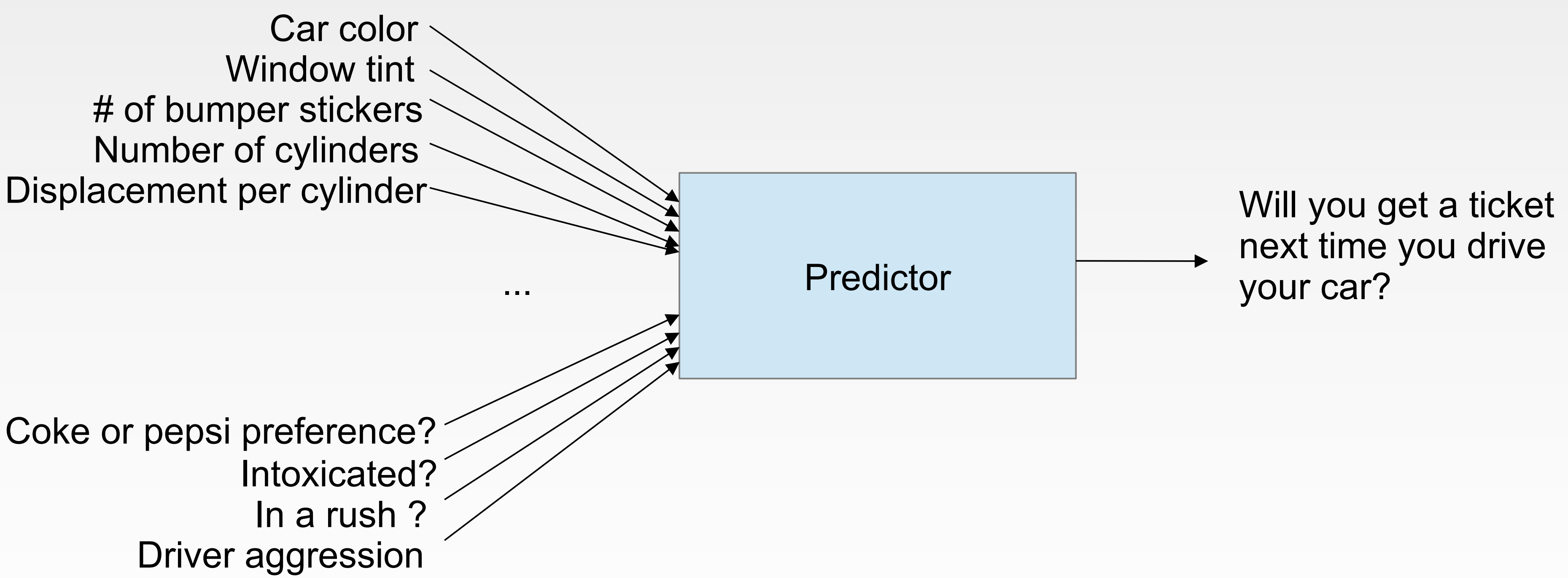
Our Graphical Model Framework:

- Reconstructability Analysis
- Representationally: sets of sets of statistically connected variables

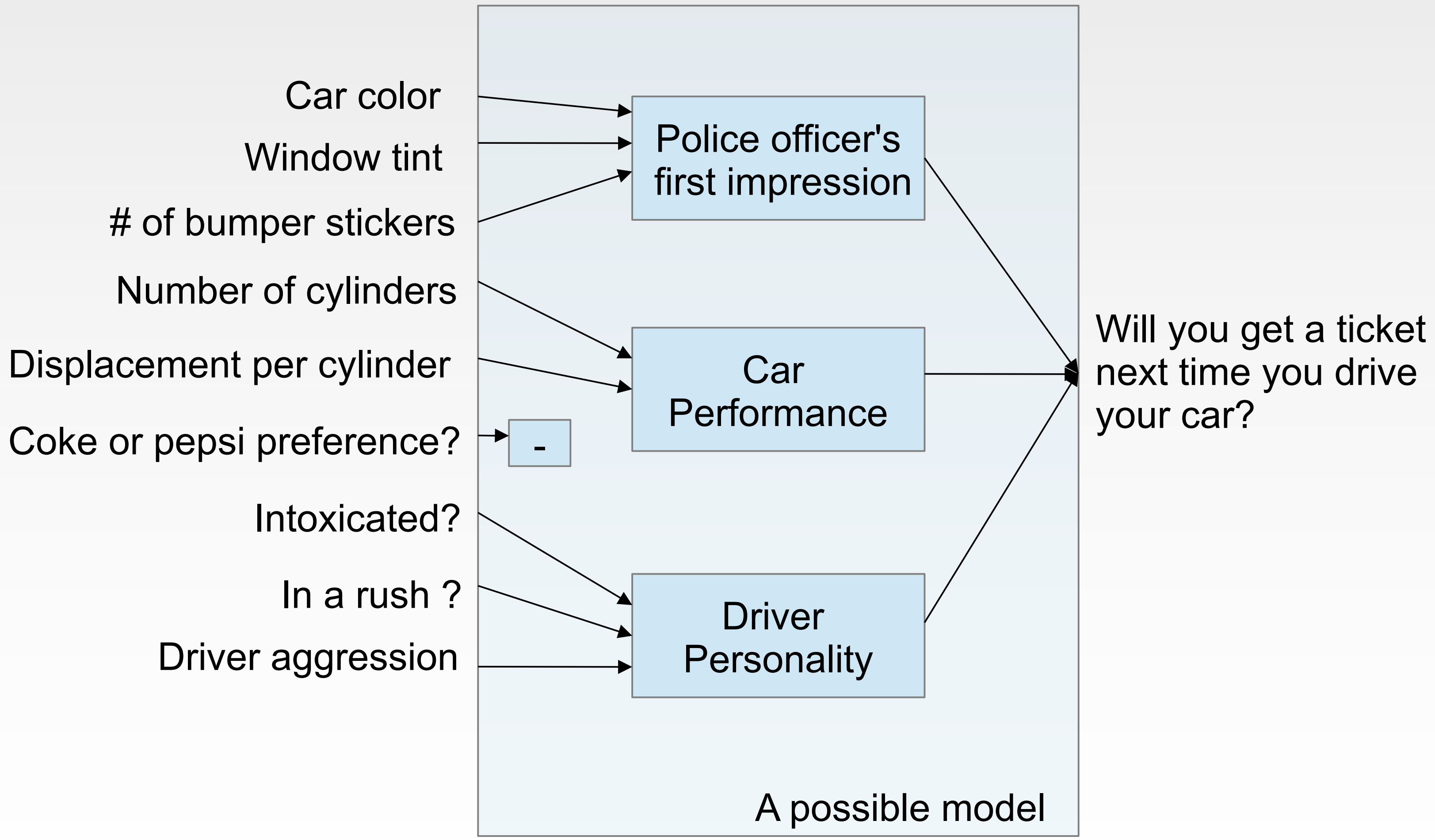
Reconstructability Analysis

- A graphical model framework with heavy overlap with Bayesian Networks and Loglinear Models.
- Uses **contingency tables to build conditional probabilities.**
- An RA model is a set of components
- Each component is a set of fully connected variables

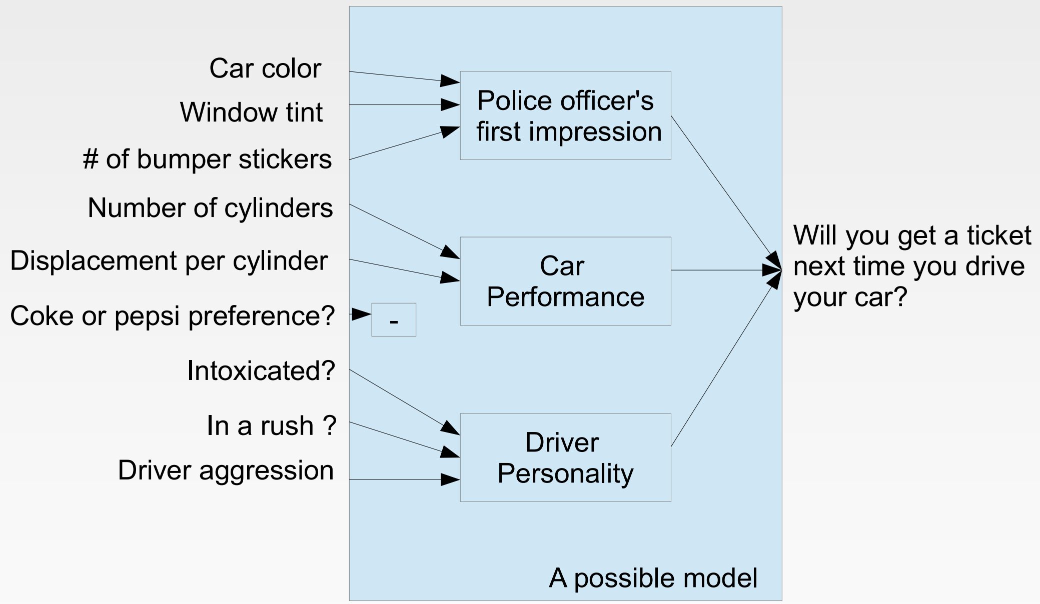
Let's Start With An Example...



Example: Underlying Structure



RA – Representation Example



Long Variable Name	Letter
Car Color	A
Window Tint	B
# of Bumper Stickers	C
Number of Cylinders	D
Cylinder Displacement	E
Coke or Pepsi?	F
Intoxicated?	G
In a Rush?	H
Driver Aggression	I
Will You Get A Ticket?	Z

Component Structure:

"First Impression" → K1 → ABCZ

"Car Performance" → K2 → DEZ

"Driver Personality" → K3 → GHIZ

Model Structure:

Model → K1:K2:K3

→ ABCZ:DEZ:GHIZ

RA – “Goodness” example

Variable Name	Letter
Car Color	A
Coke or Pepsi?	F
Ticket?	Z

Example dataset:

	F = Coke		F = Pepsi	
	A = red	A = blue	A = red	A = blue
Z=yes	**98**	**4**	**102**	**6**
Z=no	**8**	**90**	**4**	**88**

[Projections]

“AZ” Projection:

	A = red	A = blue
Z=yes	**200**	**10**
Z=no	**12**	**178**

“better”, since it contains more information

“FZ” Projection:

	F = Coke	F = Pepsi
Z=yes	**102**	**108**
Z=no	**98**	**92**

“worse”, since it looses most of the information

RA – Model Evaluation Process

Given:
- **dataset** (aggregate statistics into a contingency table holding the number of each cell's occurrence.)
- **model** (e.g., ABCZ:DEZ:GHIZ)

1. **Reduce** original data into individual contingency tables for each component

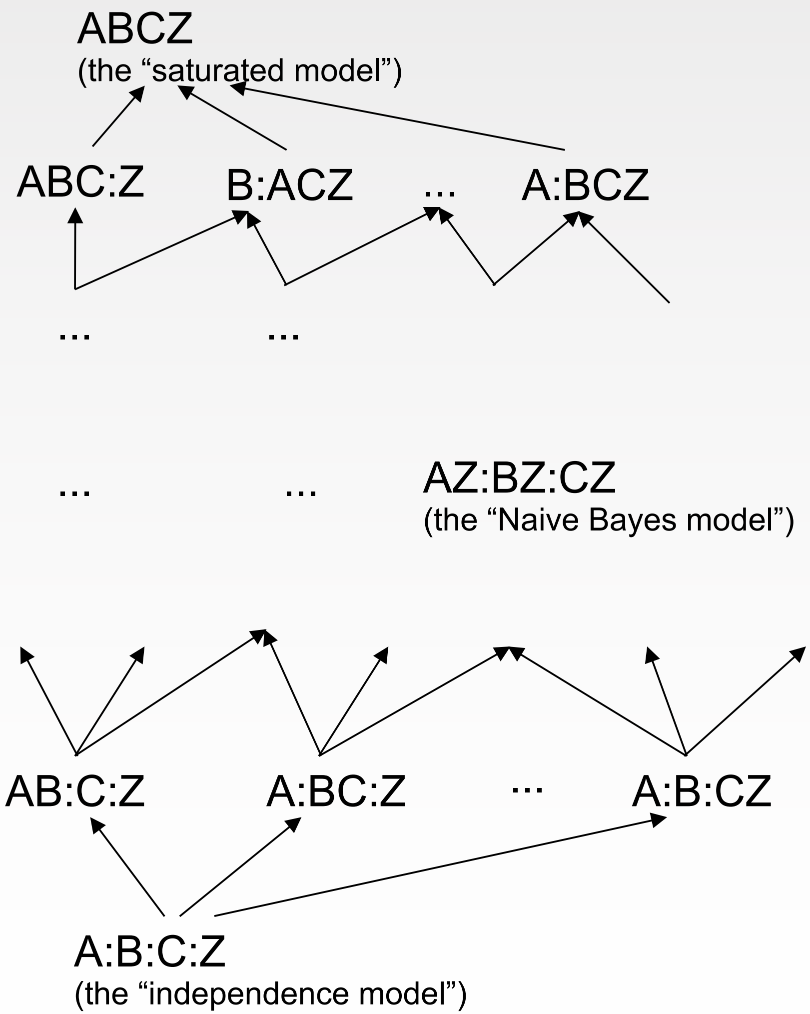
Now you have:
- **Contingency tables for each component** in the model.

2. **Construct** the dataset from the individual components.

Now you have:
- **A contingency table** in the shape of the original dataset, but respecting the constraints in the model.

3. **Evaluate** difference in terms of model complexity vs. statistical significance

RA – Lattice of Structures



Degrees of freedom (dF) increases as you climb the lattice

Likelihood (L^2) increases as you descend the lattice

Ways to balance dF and L2:

- Information Criterion:
  - **Bayesian** (BIC)
  - Akaike
  - Deviance
- Incremental alpha
- % correct on test set

Experiment 1:

- Search for a model
- Use that model as predictor in a game

Introducing: Occam

- Occam is a web interfaced tool built and maintained at Portland State University for running RA searches and evaluating RA models on arbitrary datasets.
- General information at http://dmm.sysc.pdx.edu
- Tool at http://dmm.sysc.pdx.edu/weboccam.cgi
- Note: we are currently discussing “variable based” RA.

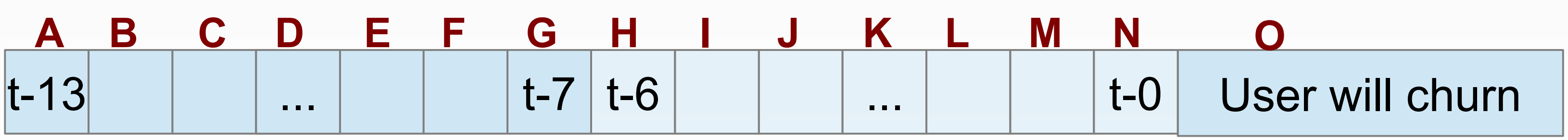
Search Results

"Loopless" Model:

Level	model	dDF	%dH(DV)	dBIC	%c (train)	%c (test)	%cover (train)	%cover (test)
4	IV:KLMNO	624	21.2287	271665.94	78.77	90.5	100	100

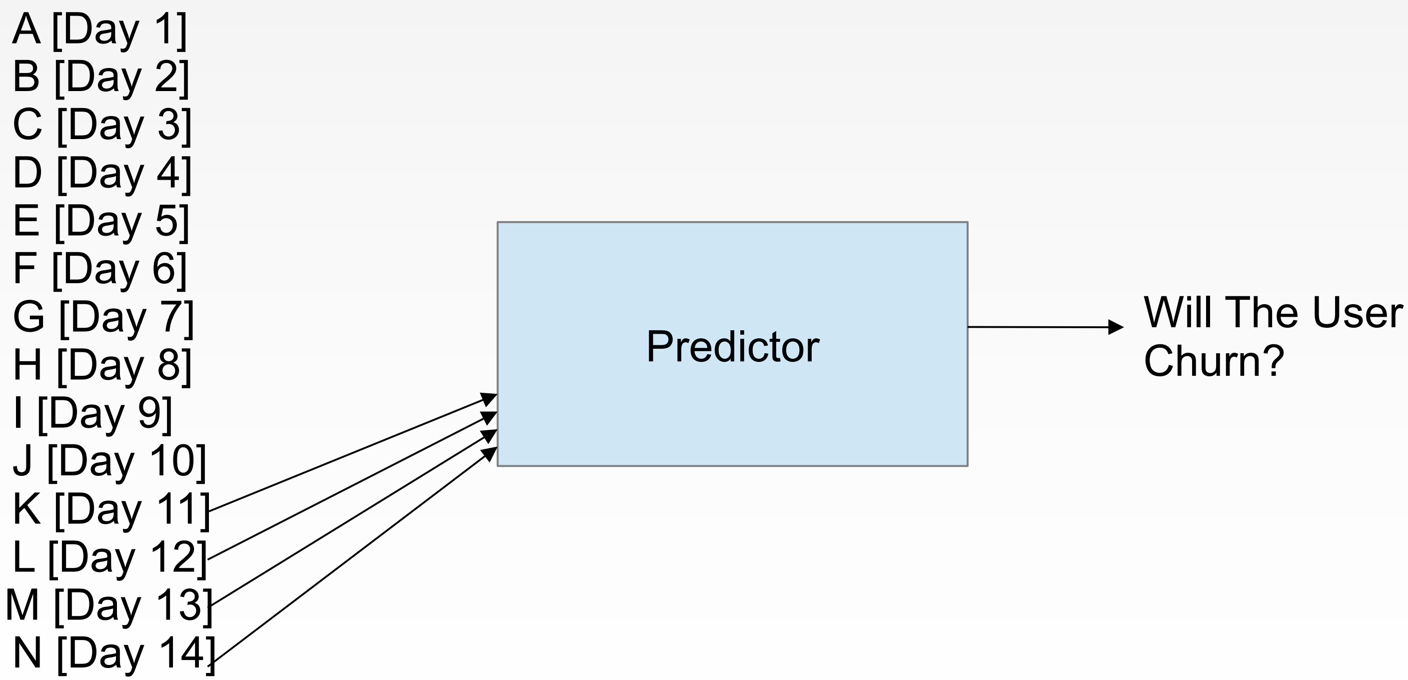
"Loopy" Model:

Level	model	dDF	%dH(DV)	dBIC	%c (train)	%c (test)	%cover (train)	%cover (test)
20	IV:BIO:BKO:CHO:CJO:HIO:ILO :JKO:KMO:LMO:LNO:MNO	212	23.5297	307800.93	79.50	87.78	79.50	76.56



Our Predictor: KLMNO

- Loopless models run way faster than loopy models



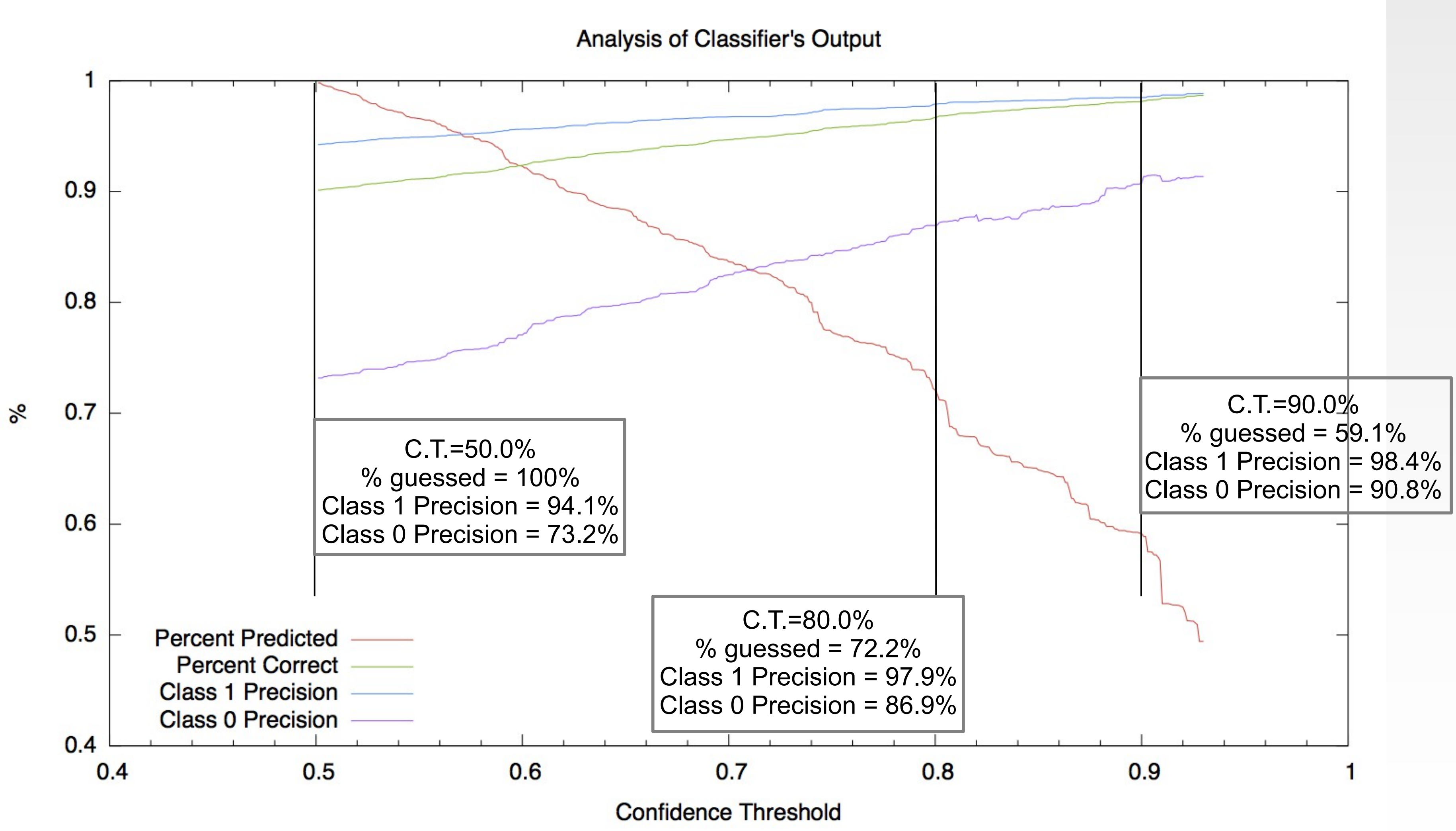
Results: Confusion Matrix

- Recall the best-by-BIC, loopless, directed model: "KLMNO"
- Train/test sets generated by picking two timesteps a month apart.

	Predicted to Churn	Predicted to Not Churn
Actual Will Churn	**818,274**	**56,481**
Actual Will Not Churn	**50,970**	**154,183**

Works out to be:
- 90.04% prediction rate,
- 93.54% will-churn precision
- 75.16% will-not-churn precision

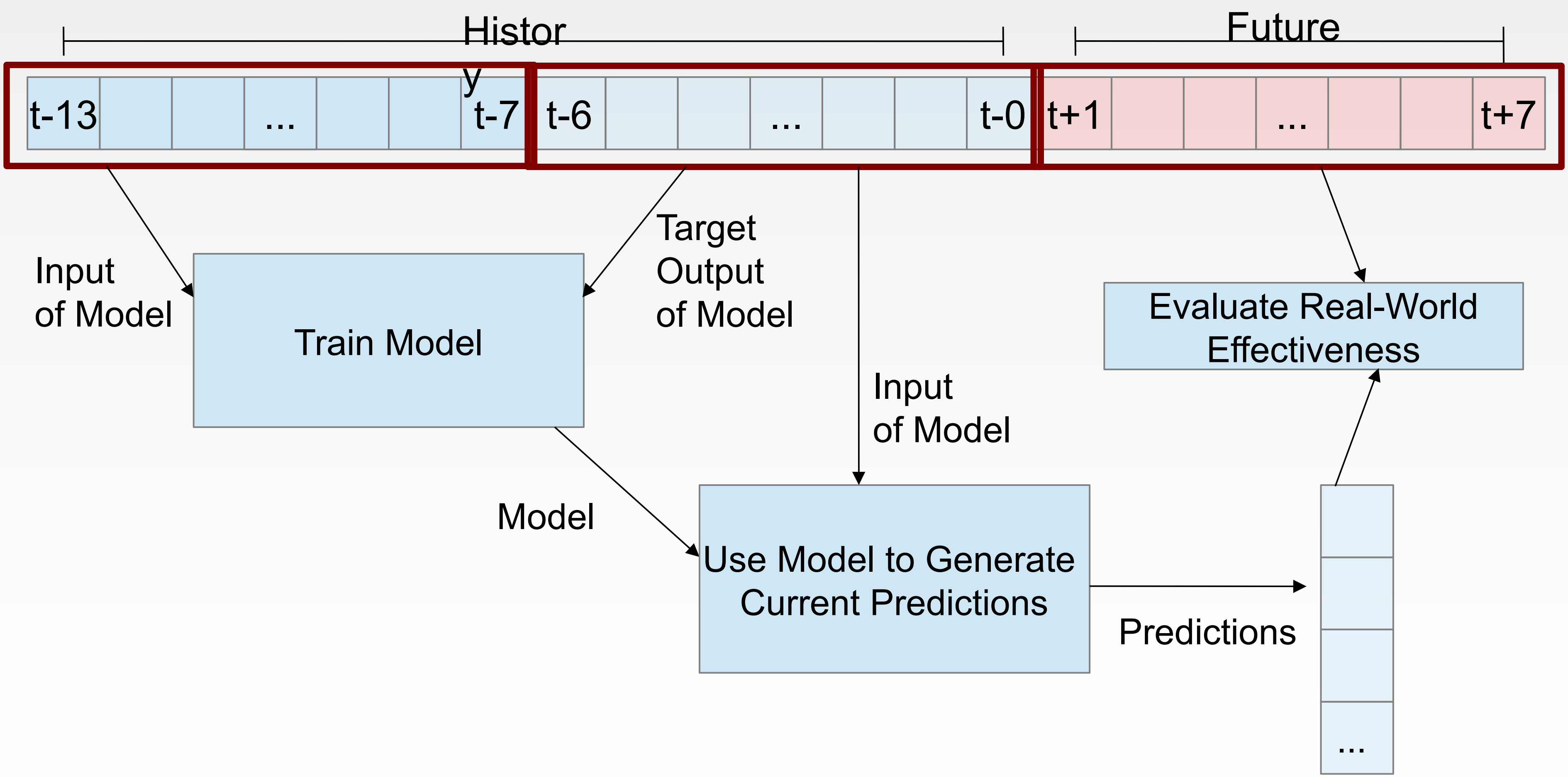
The Confidence Threshold



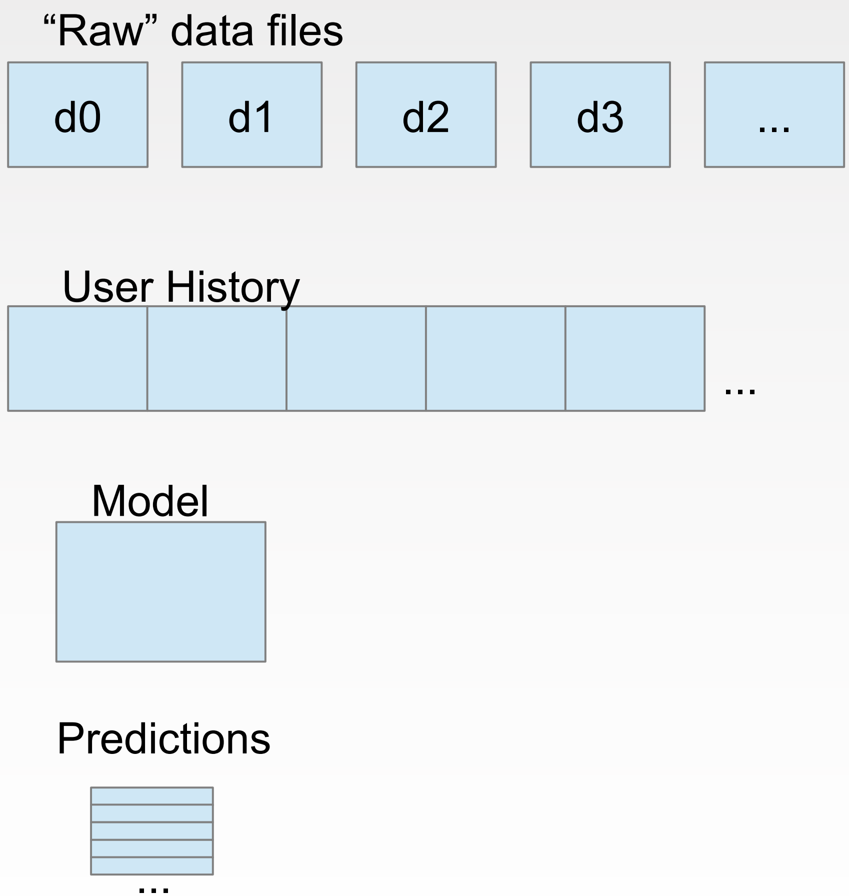
Experiment 2:

- Discuss productized implementation
- Evaluate performance on full network

Rolling Window For Model Training



Compute Time On Initialization



- (device,game,count) tuples.
- Exported from Redshift DW.
- 5 minutes per day of data.
- 11.5MM, 200Mb each

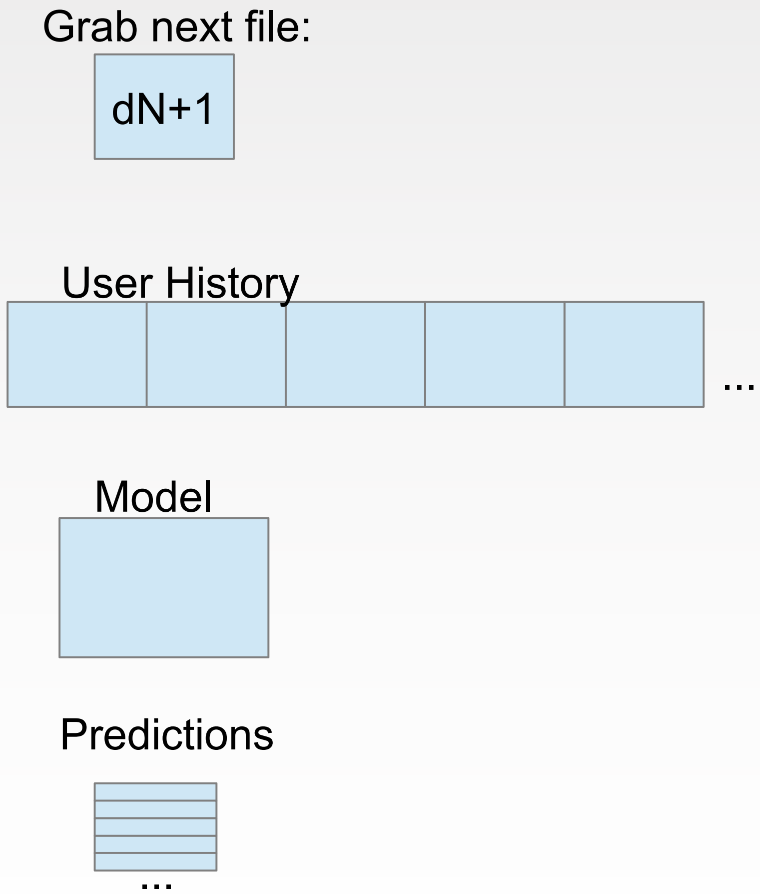
- aggregated history for each user.
- 35 – 70 minutes per day aggregated.
- 78MM rows, 7Gb

- Full contingency table
- 30 minutes
- 800 rows, 80Kb

- (device,game,prediction) tuples
- 30 minutes
- 9.5MM rows, 300Mb

- Total Processing time: around 13 hours on amazon's "m1.large" instance.

Compute Time To Iterate One Day



- (device,game,count) tuples.
- 5 minutes for one day of data.
- 11.5MM, 200Mb

Total time: about 2 hours
- backup table: 20 min
- drop last day: 35 minutes
- add new column: 10 minutes
- populate new column: 45 minutes
- 78MM rows, 7Gb

- Full contingency table
- 30 minutes
- 800 rows, 80Kb

- (device,game,prediction) tuples
- 30 minutes
- 9.5MM rows, 300Mb

- Total Processing time: around 3 hours on amazon's “m1.large” instance.

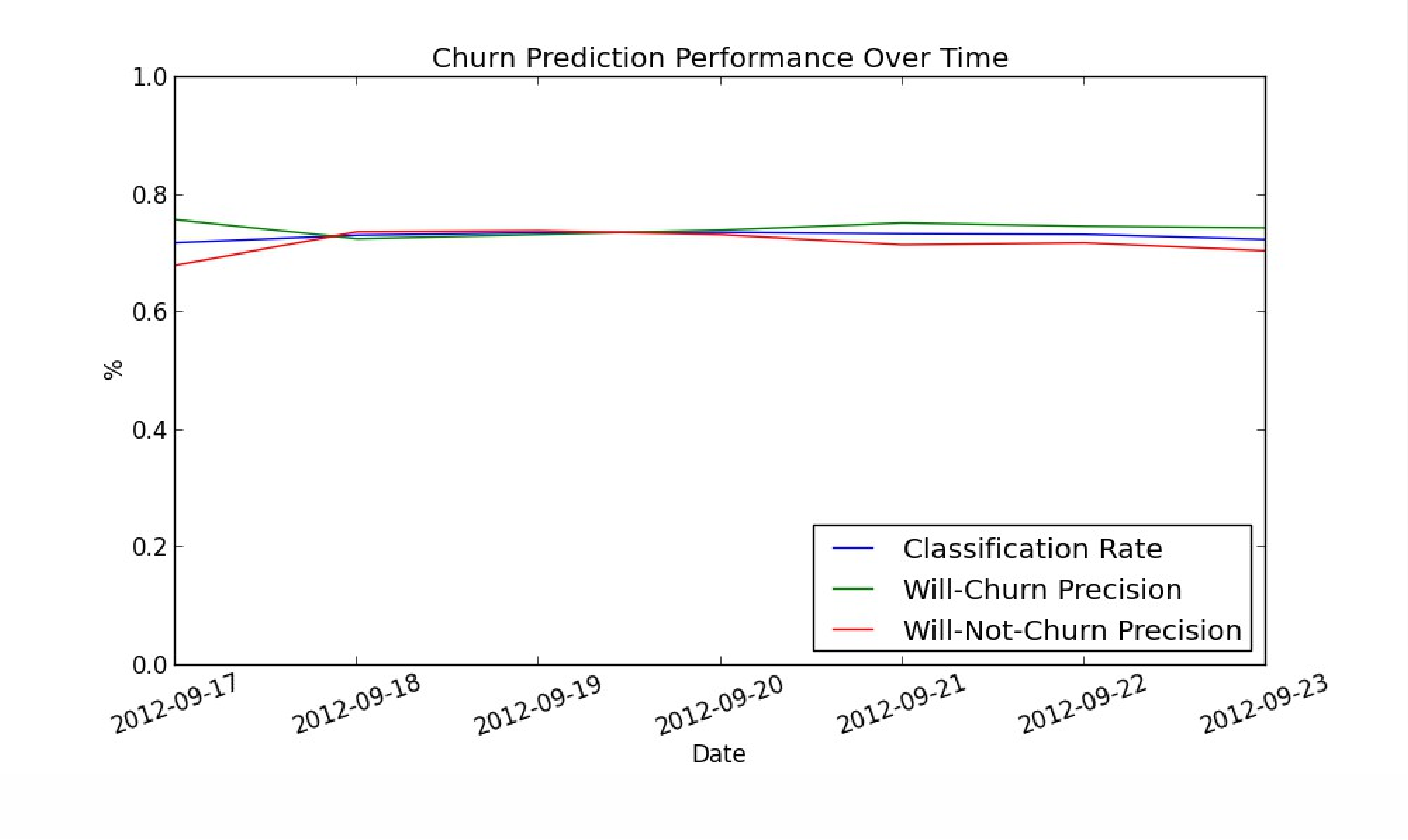
Performance On Full Network

On **2012-10-26**:
8,582,225 active device-game tuples.
Game id not considered in prediction.

	Predicted to Churn	Predicted to Not Churn
Actual Will Churn	**2,911,865**	**1,130,127**
Actual Will Not Churn	**1,120,336**	**3,419,897**

Works out to be:
- 73.78% prediction rate,
- 72.04% will-churn precision
- 75.32% will-not churn precision

Performance Across Time



Next Steps

- It works well. It's fast. ... and there's still a lot of work to do.
- To do:
  - Compare with other methods.
  - Add variables (day of week, game genre, player type, ...)
  - Evaluate performance over the last year.
  - Re-run search on full-network dataset
  - Optimize code performance