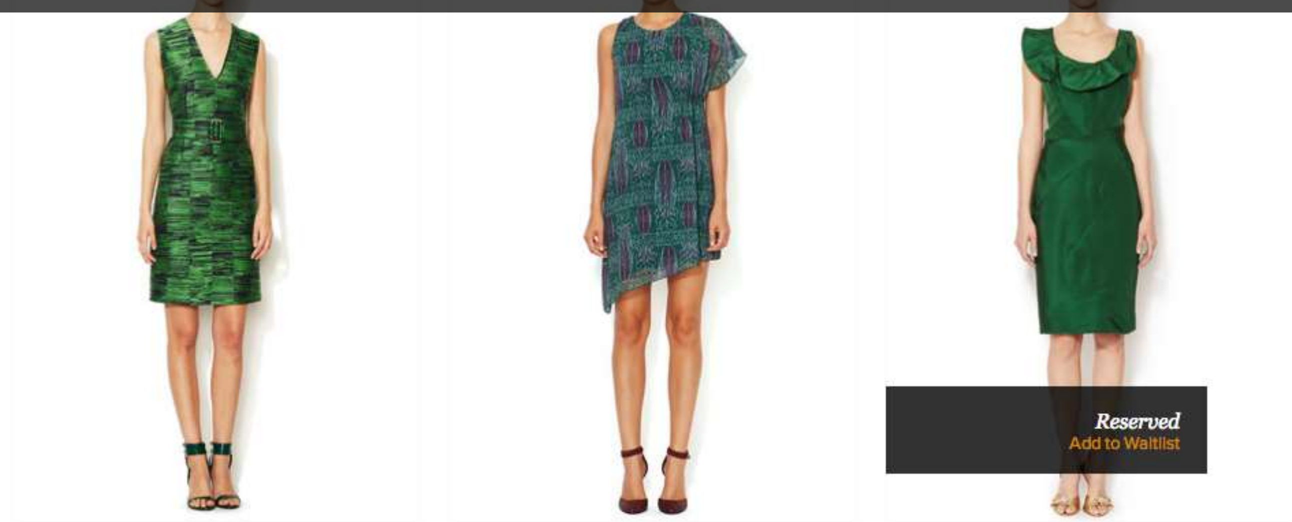
Using Machine Learning, Business Rules, and Optimization for Flash Sale Pricing

Igor Elbert, Distinguished Data Scientist, Gilt.com
Dr. Jacob Feldman, CTO, OpenRules, Inc.



Building a Pricing System for an Online Retailer

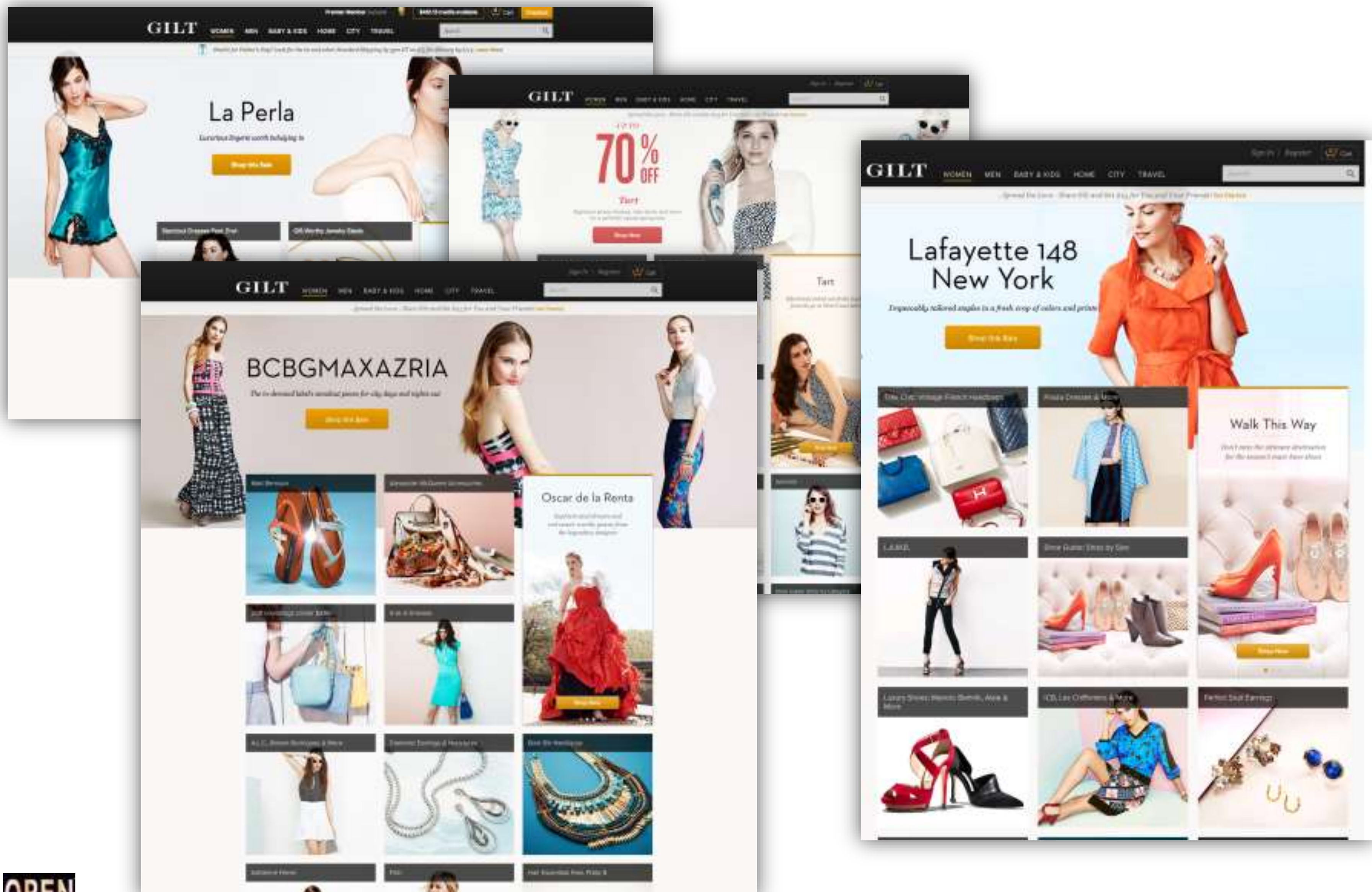
- GILT:
  - Online retailer selling curated collections of fashion products via flash sales
- Expected Functionality:
  - Utilize sales history to predict demand for ever-changing assortments of thousands of products
  - Collaborate with business domain experts to quickly generate optimal prices that can immediately go live on site

Applied Technology

- A combination of Machine Learning, Business Rules, and Multi-Objective Optimization:
  - Predictive Analytics
    - R, xgboost
  - Business Rules
    - OpenRules
  - Optimization
    - OpenRules/JSR-331 with various linear solvers

Before Gilt – sample sales

Gilt pioneered online “flash sales” in US



OPEN RULES

GILT

LIFESTYLE MARKETING PLATFORM

Gilt is a members-only lifestyle destination and ecommerce site that provides insider access to today's top designer brands as well as exclusive local experiences.

GILT

Global Reach

9.7M+

active members

400

sales launch weekly

7K+

packages shipped daily

100

countries shipped to

1M+

active mobile app users*

50%

of revenue is generated via mobile purchases

1B+

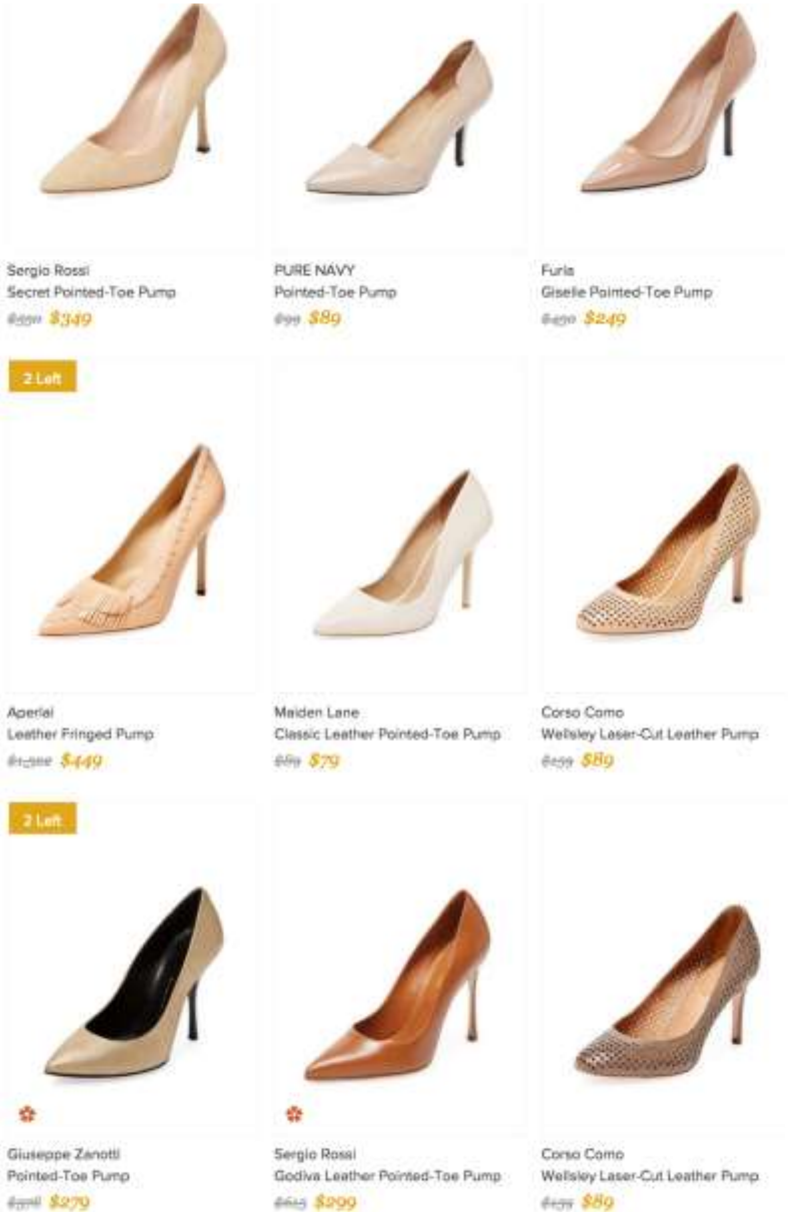
highest press impressions from a single partnership**

1.5M+

social media followers

*Gilt for iPhone, iPad & Android
**Michael Bastian x Hewlett-Packard

How to price thousands of items every day?



OBJECTIVES

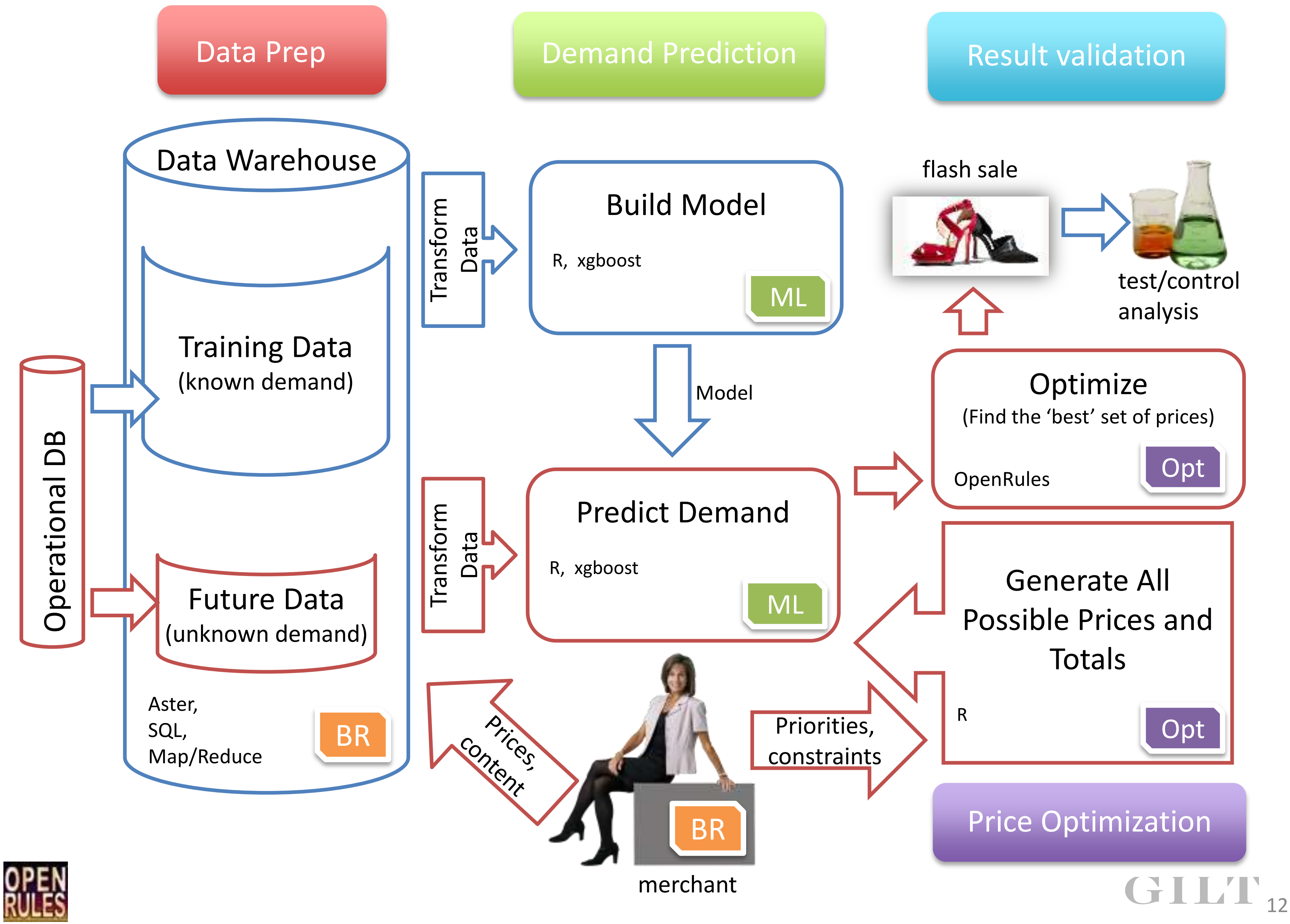
- Predict demand for every product in a given sale for all possible prices
- Find the best combination of prices to satisfy business objectives (weighted mix of revenue, margin, sell-through, etc)
- Present price recommendations to business

How it's done

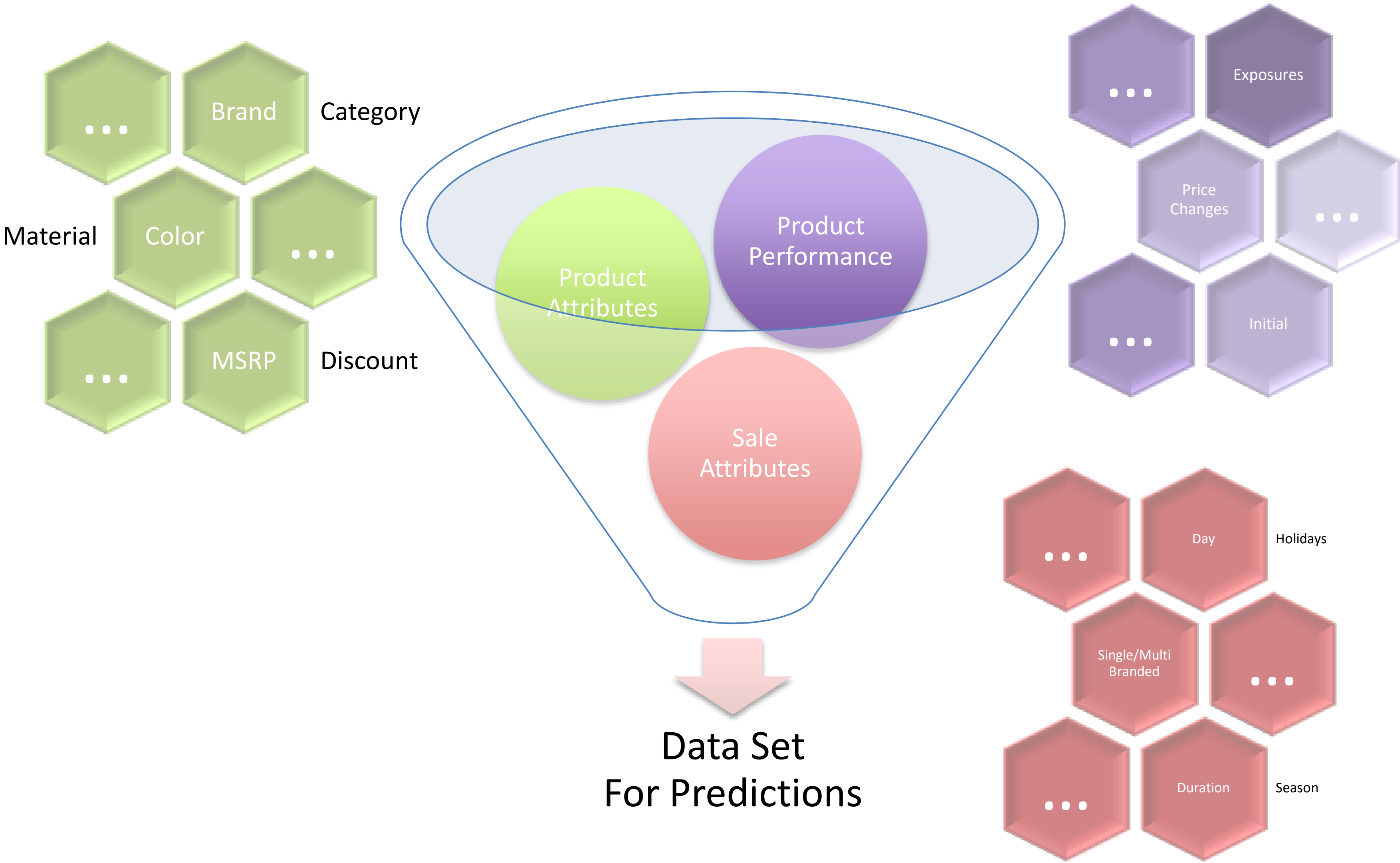
1. Data Prep
2. Demand Prediction
3. Price Optimization
4. Result validation

OPEN RULES

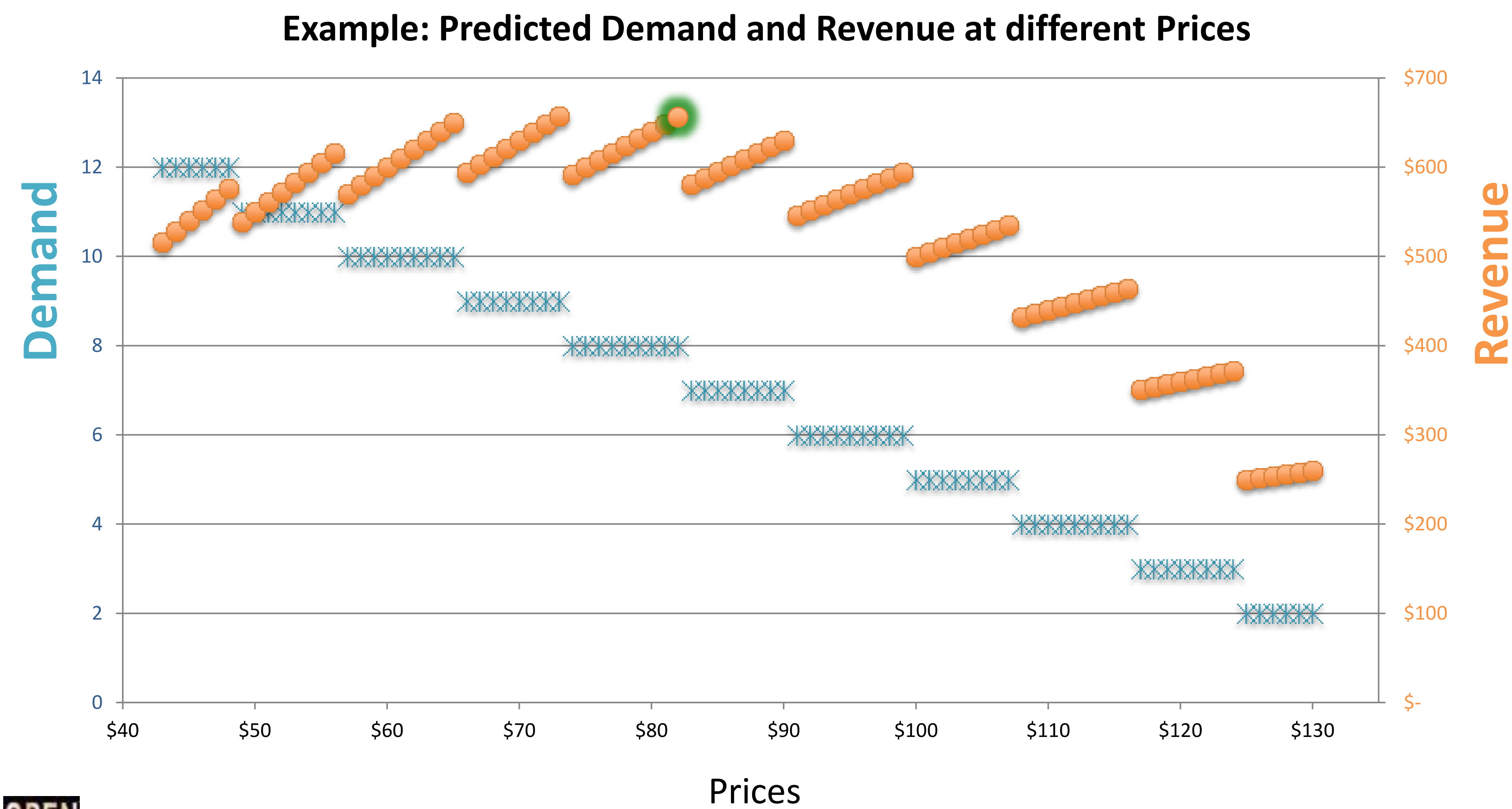
GILT


Data Prep
Demand Prediction
Result validation
Data Warehouse
Training Data
(known demand)
Future Data
(unknown demand)
Aster,
SQL,
Map/Reduce
BR
Operational DB
Transform Data
Transform Data
Build Model
R, xgboost
ML
Model
Predict Demand
R, xgboost
ML
flash sale
test/control analysis
Optimize
(Find the 'best' set of prices)
OpenRules
Opt
Generate All Possible Prices and Totals
R
Opt
Prices, content
Priorities, constraints
BR
merchant
Price Optimization
OPEN RULES
GILT

1. Data Preparation



2. Demand Prediction



3. Price Optimization

- Goals:
  - optimize per product and per sale
  - allow business user to set goals (revenue, sell-through, margin, or combination)
- Iterate quickly

Sample Rules

Minimal Number of Previous Exposures		Variable	<oper>	Value		
Is	0	**Minimum Discount from MSRP**	Is	20		Initial Sales
Is	0	**Percent Difference from Original Price**		40		
Is	0	**Minimal Margin Percent**		40		
Is	0	**Minimal Sell Through Percent**		20		
Is	1	**Minimum Discount from MSRP**	Is	20		Repeat Sales
Is	1	**Percent Difference from Original Price**		40		
Is	1	**Minimal Margin Percent**		30		
Is	1	**Minimal Sell Through Percent**		20		
Is	10	**Minimum Discount from MSRP**	Is	60		Exit Sales
Is	10	**Percent Difference from Original Price**		40		
Is	10	**Minimal Margin Percent**		5		
Is	10	**Minimal Sell Through Percent**		40		

Optimization Weights

	Variable	<oper>	Value
Gross Revenue Weight		Is	2
Gross Margin Weight			5
Gross Sell Through Weight			3

Sample Results For A Sale:

Target	Revenue	Margin	Sell-through
Max Revenue	$6,606	58%	23%
Max Margin	$4,289	67%	16%
Max Sell-through	$5,628	48%	24%

OPEN RULES

GILT

Per sale optimization

Best predictors of demand (number of units sold):

- Number of units available
- Price, Discount, MSRP
- Item price relative to the prices of other items in the sale
- Product attributes, etc

Prediction changes:
Before: predict demand for all acceptable prices
Now: same as before but for all possible totals

Example

Prices: $2 or $4

Prices: $1 or $3

Price total: $3 - $7
- Apply constraints early
- Calculate all the totals

Item	Price	Total	Demand
Ball	***$2***	***$3***	***4***
Ball	$2	$4	4
Ball	***$2***	***$5***	***4***
Ball	$2	$6	3
Ball	$2	$7	3
Ball	$4	$3	2
Ball	$4	$4	2
Ball	***$4***	***$5***	***2***
Ball	$4	$6	2
Ball	***$4***	***$7***	***2***
Pen	***$1***	***$3***	***7***
Pen	$1	$4	7
Pen	***$1***	***$5***	***8***
Pen	$1	$6	8
Pen	$1	$7	8
Pen	$3	$3	1
Pen	$3	$4	1
Pen	***$3***	***$5***	***1***
Pen	$3	$6	1
Pen	***$3***	***$7***	***0***

GILT

Multiple Knapsack Problem / Bin-packing problem

- All items must be priced
- Each item must have only one price
- Sum of all prices should equal to one and only one total

OPEN RULES

GILT

Problem definition (MathProg)

```
set Look;
set Price := 1..10000;
set Total := 1..100000;

set Look_Price_Total within {l in Look, p in Price, t in Total};
param price  {(l,p,t) in Look_Price_Total}, >= 0, integer := p;
param demand {Look_Price_Total}, >= 0, integer;
param revenue{(l,p,t) in Look_Price_Total} := price[l,p,t] * demand[l,p,t];

param orig_price  {Look_Price_Total}, >= 0, integer, default 0;
param base_price  {Look_Price_Total}, >= 0, integer, default 0;
param msrp_price  {Look_Price_Total}, >= 0, integer, default 0;
param num_units_available {Look_Price_Total}, >= 0, integer, default 0;

set Unique_Total := setof{(l,p,t) in Look_Price_Total} t;

var Use {Look_Price_Total} binary;
var Use_Total {Unique_Total} binary;

maximize Revenue:    sum{(l,p,t) in Look_Price_Total} revenue[l,p,t] * Use[l,p,t];

s.t. one_of_each{l in Look}: sum{(l,p,t) in Look_Price_Total} Use[l,p,t] = 1;
s.t. single_total: sum{t in Unique_Total} Use_Total[t] = 1;
s.t. price_sum_is_total{t in Unique_Total}:
        sum{(l,p,t) in Look_Price_Total} price[l,p,t] * Use[l,p,t] = t * Use_Total[t];
```

```
set Look := Ball Pen;
param: Look_Price_Total: demand :=
Ball    2   3   4
Ball    2   4   4
Ball    2   5   4
Ball    2   6   3
Ball    2   7   3
Ball    4   3   2
Ball    4   4   2
Ball    4   5   2
Ball    4   6   2
Ball    4   7   2
Pen     1   3   7
Pen     1   4   7
Pen     1   5   8
Pen     1   6   8
Pen     1   7   8
Pen     3   3   1
Pen     3   4   1
Pen     3   5   1
Pen     3   6   1
Pen     3   7   0;
```

GILT

Modeling and Solving Real-world Problems

- We modeled the problem using OpenRules and JSR-331 Standard
- Real optimization problems consist of hundreds of thousands records:
    - We used JSR-331 Constraint Solvers to validate the problem correctness. But actual problems were too large for constraint solvers
    - We tried various JSR-331 Linear Solvers (GLPK, LP-Solve, COIN suite, SCIP, and others)
    - None was able to solve large problems in a reasonable time or at all

How We Solved the Production Problem

- OpenRules was able to create a rules-based decision model that automatically splits one large problem into a set of smaller sub-problems (one for every individual total cost)
- While there may be thousands of sub-problems, JSR-331 Linear Solvers are able to quickly solve them
- Then OpenRules decision model analyzes all found solutions to come up with the optimal solution that satisfy a configurable combined objective – a maximal combination of Revenue, Margin, and Sell-Through
- Big advantage of this approach: it can be parallelized to solve even much larger problems!

Conclusion

- We applied a combination of Machine Learning, Business Rules, and Multi-Objective Optimization to solve a real-world operational problem – flash sale price optimization
- The pricing methodology and tools that support each of these 3 decision management techniques were readily available and quite powerful
- However, the production-level problems required a special ingenious approach to actually solve these problems

Questions?

GILT

Igor Elbert
ielbert@gilt.com

Jacob Feldman
jacobfeldman@openrules.com