Prescient is developing next gen ML capabilities for clients to answer meaningful questions around demand forecasting, spend optimization, and customer lifetime value.

However, to effectively scale rev ops, Prescient needs to streamline data engineering across clients to accelerate time to value.
Prescient Early Days

Data

Model
Every Customer is Unique

Customer 1:
- 20 Data sources
- 10 columns each
- 12 hours of maintenance a week

Customer 2:
- 14 Data sources
- 12 columns each
- 50% in spreadsheets

Customer 3:
- 8 Data sources
- 2 columns each
- 8 hours of maintenance a week

We’re data scientists and also, DBAs, Sys-admins...
THE CHALLENGES DISCUSSED

Prescient Features Hold Predictive Value but are not Accessible in a Single API

Lack of automated management of features across Prescient customers

Lack of streamlined feature replication across similar data schemas

Lack of scalable centralized repository for storing and viewing Prescient features
Focus On Efficiency In Order To Grow

Operationalize labor-intensive feature engineering work

Deliver models faster with greater accuracy

Less time debugging different versions of one-off scripts

Speed up onboarding new customers
THE SOLUTION

A Feature Store for Standardization & Acceleration

CENTRALIZE
Features across multiple customers

AUTOMATE
Production data and modeling pipelines for data scientists

ADOPT
Visual interface to evaluate, govern, and adopt outcomes
1. Ingest

2. Transform

3. Train/Deploy

Multi-Stage ML Pipeline

ETL/ Feature Eng.

PyRasgo

Model Training

Fivetran

snowflake

Feature Set

S3

CLV Model

Logging Data
Rasgo Feature Engineering Platform

**UNDERSTAND**
- Feature Metadata
- Source and Feature Lineage
- Feature Profiles
- Feature Importance

**PREPARE**
- User Defined Transforms (UDTs) orchestrated by Rasgo
- Pre-built functions for feature engineering
- Auto-join your features from multiple sources

**SERVE**
- Serve features to models for both training and inference with PyRasgo
- Track feature drift and feature metadata changes over time
- Natively integrate with production ML ops pipelines
Rasgo Transformers

- User-defined transforms are templated SQL functions for Rasgo Objects
- User-defined transforms are written in SQL but accept python arguments via PyRasgo
- Expanding library of predefined transformations
- Designed to be shared across teams and projects

```python
t1 = li_source.transform(
    transform_name='new_lag',
    Columns = ['COST_IN_USD','CLICKS'],
    Amounts = [1,2,3,7],
    OrderBy = 'DAY',
    Partition = 'CAMPAIGN_ID'
)```
A FEATURE STORE IS NOT A SEPARATE DATA WAREHOUSE

Duplicate infrastructure and data lead to high cost of ownership and horrendous user experience. The answer is ELT.

140X Reduction in Cost to Compute
17X+ Faster Feature Query Performance

30 minutes to deploy on Snowflake
Dev features are immediately prod ready
Thank you!

Do you have any questions?

Patrick Dougherty
Co-Founder and CTO
patrick@rasgoml.com

Cody Greco
Chief Technology Officer