Leveraging Apache Beam for Enhanced Financial Insights at Credit Karma

Beam Summit 2025

Our Mission: championing financial progress for everyone

- Credit Karma is a personal finance company dedicated to helping people feel more confident about their finances.
- We provide tools and insights to help members understand and improve their financial health.
- **Key challenge**: Delivering personalized insights and product recommendations requires understanding billions of user transactions and external data.
- **Scale**: Processing 10-100 terabytes of data from various sources daily.
- **The goal**: Turning raw financial data into actionable, real-time advice.

Agenda

Data Ingestion

Collection, validation, and initial processing of raw data from various sources

Presented by: **Naresh**

Feature Engineering

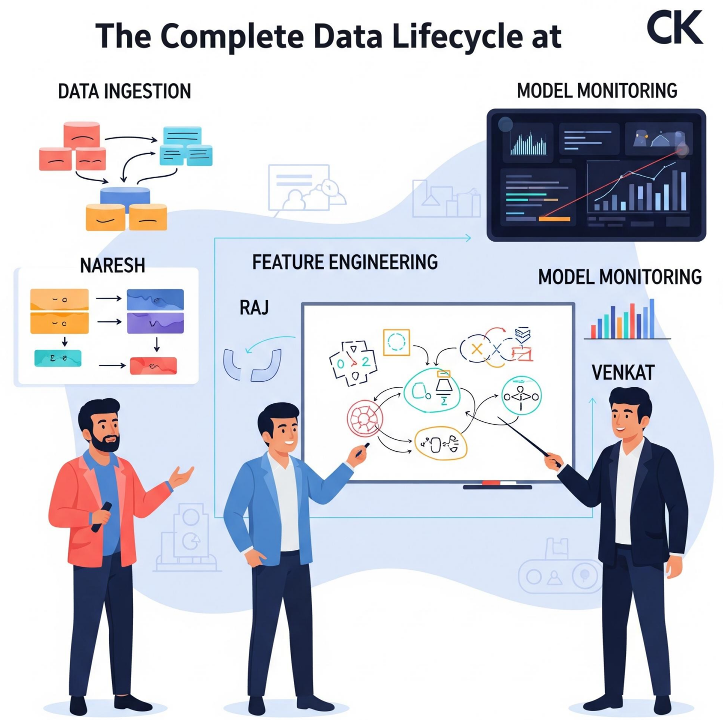
Transformation of raw data into model-ready features through preprocessing and generation

Presented by: **Raj**

Model Monitoring

Post-deployment surveillance of model performance, drift detection, and maintenance

Presented by: **Venkatesh**



Data Ingestion

Advanced Data Pipeline Solutions: Hydration,ETL and Archiving using Apache beam

This presentation explores three critical data engineering use cases: real-time data hydration, Spanner to BigQuery ETL, and high-volume GCS archiving. We'll examine the challenges, architecture patterns, and performance optimizations for each scenario to help you implement robust data pipelines at scale.

Use Case 1: Data Hydration Implementation

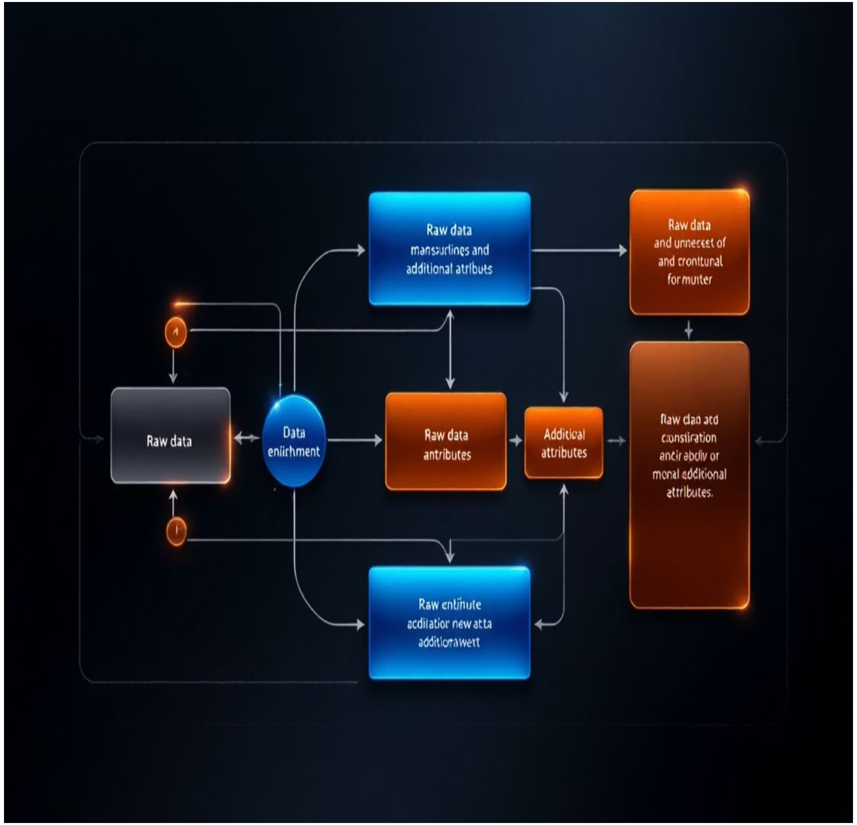
Overview and Challenges

What is Data Hydration? A process that enriches raw data by adding supplementary context or metadata for downstream systems through:

- Joining with reference data
- Data cleaning and normalization
- Schema validation and transformation

Key Challenges:

- Data Integration: Joining large datasets with external references across diverse formats
- Performance Bottlenecks: Latency issues in real-time enrichment scenarios
- Error Handling: Managing enrichment failures due to invalid/missing reference data



Data Hydration: Solution Architecture

Apache Beam Pipeline

Implement unified batch and streaming pipelines using Apache Beam's programming model

- Stream: Real-time enrichment from Pub/Sub and Kafka
- Batch: Hydrate data using cache layers or external systems

Reference Data Integration

Use side inputs for efficient enrichment operations

- External data stored in BigQuery, Spanner, or Cloud Storage
- Apply ParDo or MapElements for transformations

Performance Optimizations

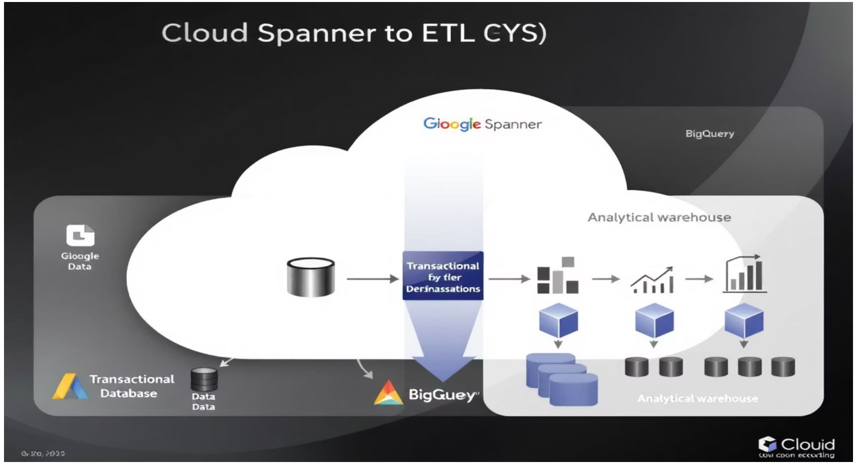
Implement techniques to minimize latency and maximize throughput

- Cache reference data to prevent redundant reloads
- Batch windowing to reduce compute costs
- Configure dead letter queues for error handling

Use Case 2: Spanner to BigQuery ETL

Overview and Challenges

ETL Purpose: Extract, Transform, and Load pipelines from Cloud Spanner (OLTP) to BigQuery (OLAP) for analytical processing.



Key Challenges:

Scalability
Extracting large Spanner datasets across multiple nodes while maintaining query efficiency for high read throughput

Schema Transformation
Aligning Spanner's normalized schema to BigQuery's denormalized analytical model

Cost Management
Optimizing compute and query costs during extraction and loading phases

Security
Docker based flex template
IAM access controls

ETL Context:

Input: Extract records from Cloud Spanner (batch or streaming)

Process: Transform and aggregate data for analytics

Output: Load processed data into BigQuery for OLAP workloads

Spanner to BigQuery: Solution Architecture

Dataflow Pipeline Components

```
s"Read $tableName in batch",
SpannerIO
  .read()
  .withSpannerConfig(sourceConfig.buildSpannerConfig)
  .withTable(tableName)
  .withColumns(TableField.toReadColumns(tableName, sourceConfig.tableFields).asJava)
  .withTimestamp(sourceConfig.snapshotReadTimestamp),
```

Batch vs. Streaming:

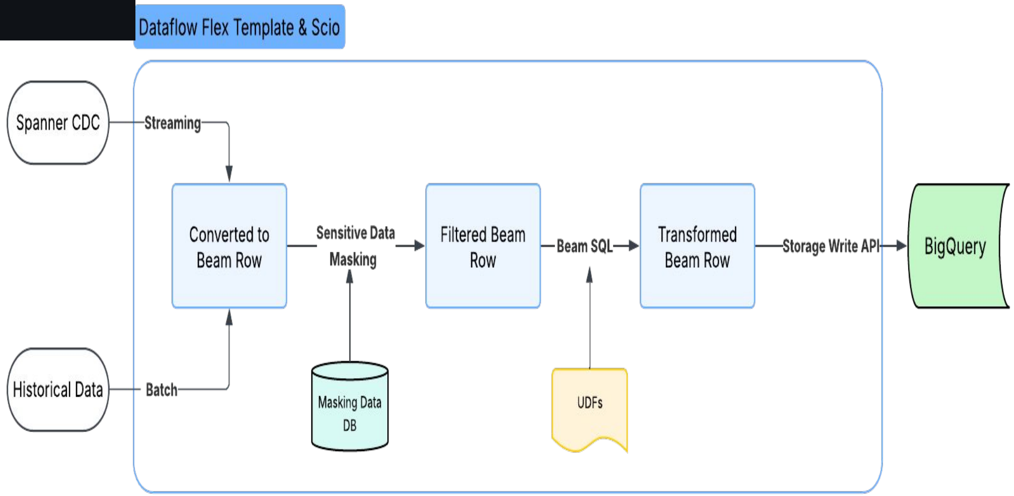
- Batch Mode: Historical data migration
- Streaming Mode: Real-time pipelines for immediate updates using spanner and Bigquery CDC

Optimization Techniques

Partitioned Extract: Split large Spanner queries into smaller time-based batches for batch processing

Denormalized Schema: Flatten Spanner interleaved tables for faster analytical queries

Cost Control: Auto Scaling



Use Case 3: GCS Archiver (100TB Daily)

Overview and Challenges

Goal: Archive 100TB of data daily in Coldline Storage using gzip compression for long-term retention.

Key Challenges:

Compression Performance

Scaling pipelines to compress 100TB daily without introducing processing delays

Storage Costs

Optimizing compression ratios to reduce storage size and minimize transaction costs

Pipeline Efficiency

Handling bottlenecks due to file I/O operations and balancing worker resources

Archival Context:

Input: Raw files from GCS (txt, json, csv)

Process: Apply gzip compression

Output: Compressed .gz files in Coldline Storage

GCS Archiver: Solution Architecture

Dataflow Pipeline Design

Read: Input files from GCS source buckets

Compress: Apply parallel gzip transformations

Write: Save .gz files to Coldline Storage

```
output_path = f"gs://coldline-bucket/{file_name}.gz"# Create
bucket with Coldline Storage class:gsutil mb -c coldline -l
us-central1 gs://coldline-bucket/
```

Performance Optimizations

Gzip Compression: Reduce file sizes from 100TB → 30TB daily (~70% savings)

Parallelization: Enable autoscaling in Dataflow

```
--autoscalingAlgorithm=THROUGHPUT_BASED
--maxNumWorkers=50
```

Batch Processing: Aggregate small files before compression

Key Takeaways & Next Steps

Velocity

Faster time to market due to cloud native

Easier to manage

Built in monitoring

Dataflow offers built in metrics and logs which saves developer time

Cost Savings

AutoScaling

Network costs

Feature Engineering: Hybrid Event-Based Aggregation for Low-Latency Fraud Prevention

Feature Engineering: Transforming Data into Intelligence

- Credit Karma's ML Landscape & The High-Stakes Fraud Use Case
- Challenges in Aggregation Strategies
- The Hybrid Solution with Apache Beam
- Architecture Deep Dive
- Results & Key Lessons

The High-Stakes World of Fraud Prevention

The challenge: Financial fraud detection requires a system that is both:

- Fast: Decisions in milliseconds to not impact user experience.
- Accurate: Minimize false positives (angry customers) and false negatives (lost money).

Our Core Requirements:

- Latency: < 100ms at p99
- Throughput: 100+ TPS
- Accuracy: > 99%
- Non-Blocking: Fraud checks cannot delay core transaction services.

Why Apache Beam? Our Strategic Platform Choice

The Pre-Beam Dilemma: Immense engineering investment and resources with a slow time-to-market.

The Solution Criteria: We needed a platform that offered:
- Robustness & Scalability
- Faster Time to Market (with minimal re-platforming effort)
- Optimal Investment (getting max value for resource allocation)
- Consistency across batch and streaming.

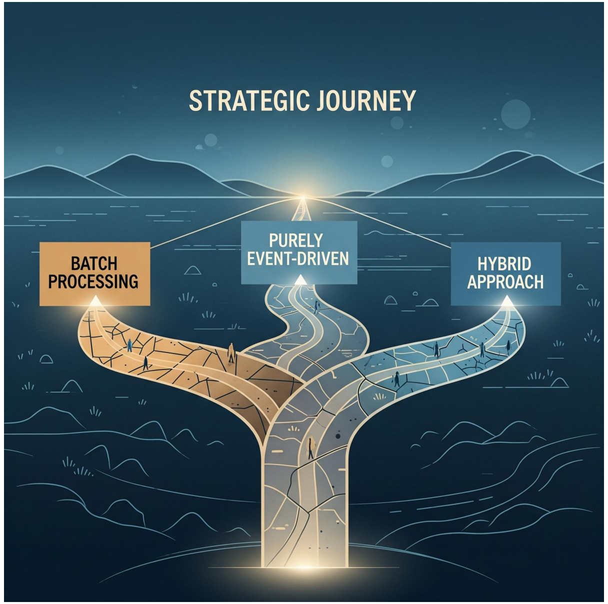
Apache Beam & Dataflow Fit:
- Managed Service (Google Dataflow): Drastically reduced operational burden & infrastructure investment.
- Proven Scale & Reliability: Cloud-native architecture built for our needs.

The Journey: choosing the right aggregation strategy

To solve this, we had to go back to first principles. Any data processing logic is triggered in one of two fundamental ways:

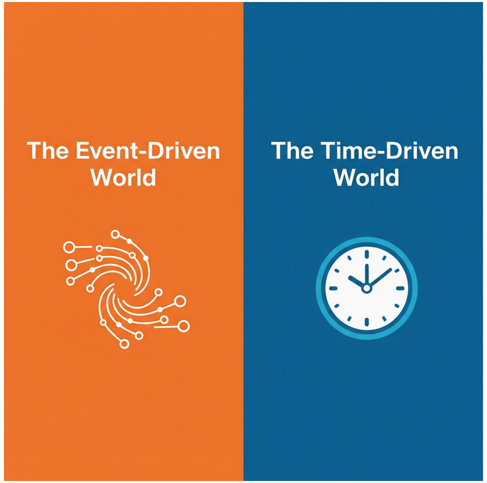
- When new data arrives.
- When the clock ticks.

A Hybrid Approach: The one that actually worked for us.



The Two Worlds of Processing Triggers

	The Event-Driven World	The Time-Driven World
Trigger	When new data arrives.	When the clock ticks.
Strength	Minimal Latency.	Logical Correctness.
Weakness	Fails on inactivity(Stale State).	Introduces latency.



Our challenge: Get the speed of the Event-Driven world with the correctness of the Time-Driven world

Why "Pure" Event-Driven Isn't Enough

A purely event-driven architecture reacts to incoming events.
But what happens when there are no events?

Problem 1: Inactivity & Stale State

- Aggregations are only triggered by new events.
- If a user becomes inactive, their state (e.g., "high transaction velocity") can become stale and never reset.

Problem 2: Orthogonal Pipelines

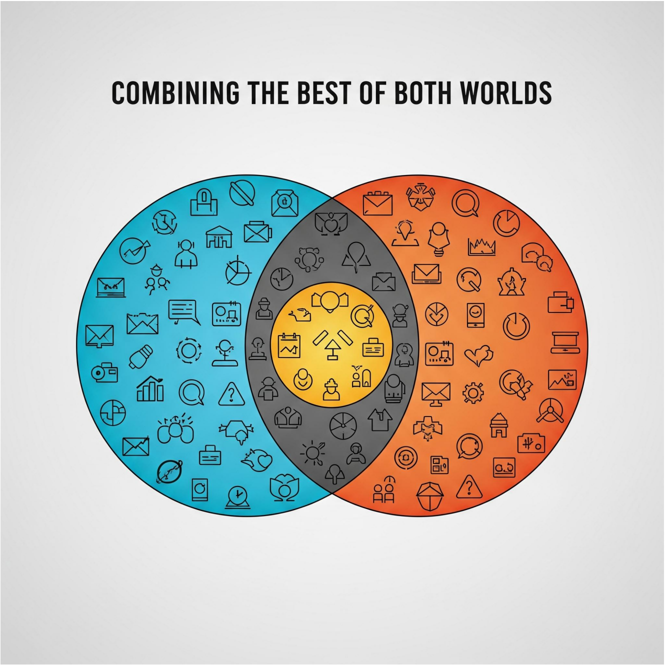
- External systems (like a rule engine) might query the user's state.
- If the state is stale, the external system gets incorrect data, leading to bad decisions.

The Solution: A Hybrid of Event-Driven and Time-Based Logic

The Hybrid Idea: Combine the best of both worlds.

- Event-Driven Processing: React instantly to new events for minimal latency.
- Time-Based Guarantees: Use timers to ensure calculations are finalized or reset, even with no new events.

This is a perfect use case for Apache Beam's State and Timer APIs.

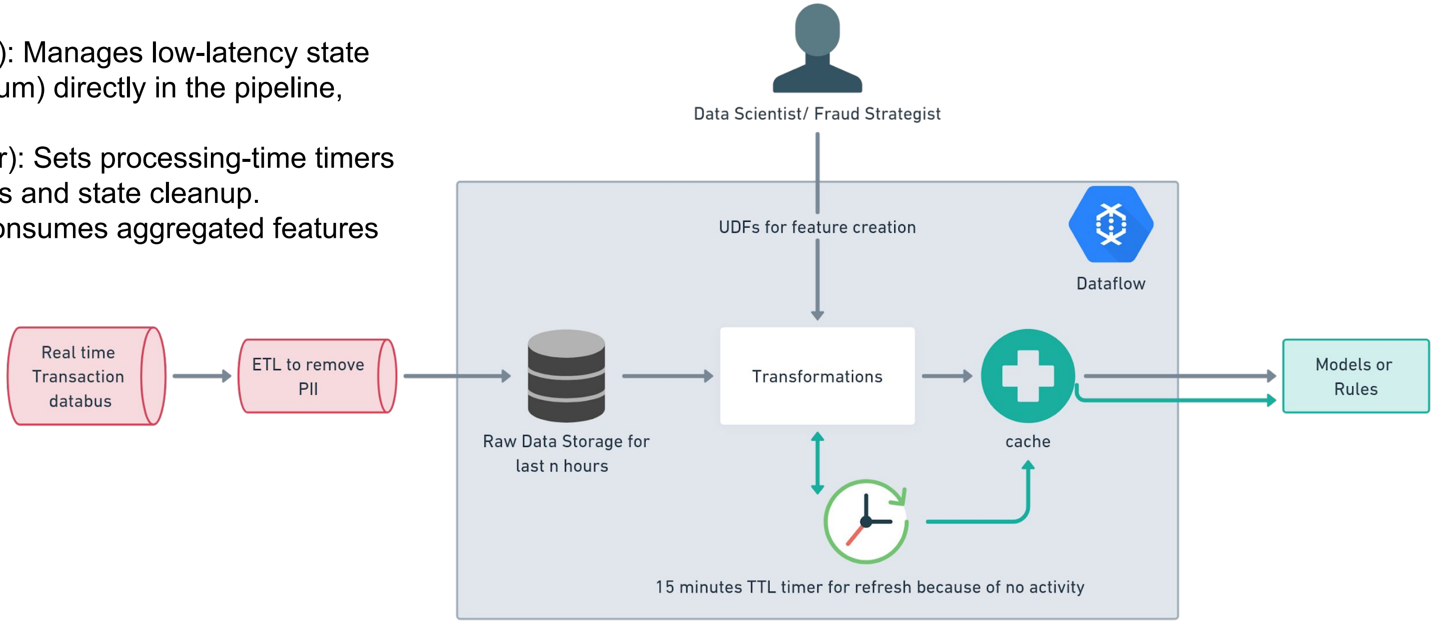


Our Hybrid Architecture on Google Cloud

Ingestion (Pub/Sub): Scalable, reliable event stream.
Processing (Apache Beam / Dataflow):

- Stateful DoFn: The core of our logic, holding per-user aggregations.
- Beam State API (@State): Manages low-latency state (e.g., transaction count/sum) directly in the pipeline, backed by Windmill.
- Beam Timer API (@Timer): Sets processing-time timers to trigger final calculations and state cleanup.

Detection (Fraud Engine): Consumes aggregated features for ML model scoring.



The Results: A Decisive Win for the Hybrid Approach

- Latency Achievement: Features delivered at <100ms at p99
- Throughput Achieved: Successfully processes at 100+ Transactions Per Second (TPS) with high consistency.
- Business Impact: Significant Reduction in Fraud Losses
- Operational Efficiency: Managed services and streamlined architecture.
- Accelerated Innovation: Engineers now focus on new fraud-detection capabilities, not infrastructure, leading to faster time-to-market for new models.

Monitoring Model and Feature Drift with Dataflow

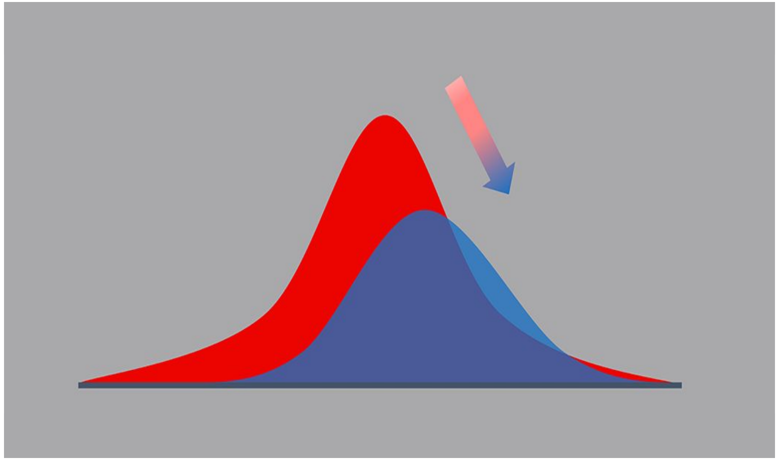
What is Model Monitoring?

What is Model Drift?

- The natural degradation of a model's predictive accuracy over time. It happens when the real-world data a model sees in production no longer matches the data it was trained on.

The Two Core Types of Drift

- **Concept Drift:** The relationship between what you're measuring and the outcome changes. The meaning of what you're predicting has shifted.
  - ***Example:*** A movie recommendation system stops working well after a user moves to a new country and their viewing preferences shift
- **Data Drift:** The statistical properties of your input data change. The population your model is scoring is fundamentally different now.
  - ***Example:*** A weather prediction model sees new patterns because it's now summer instead of winter, altering the data distribution but not the meaning of the outcome



Our Solution: A Proactive Strategy

What is Model Monitoring?

1. **Detection:** We use continuous monitoring with statistical tests (PSI/CSI) to automatically alert us when data drift occurs.

PSI Value	Interpretation
< 0.1	Stable distribution; no action needed.
0.1 to < 0.25	Minor shift; investigation recommended.
≥ 0.25	Significant shift; model may require retraining or updating.

Bins/buckets

Baseline

Comparison/prod Data

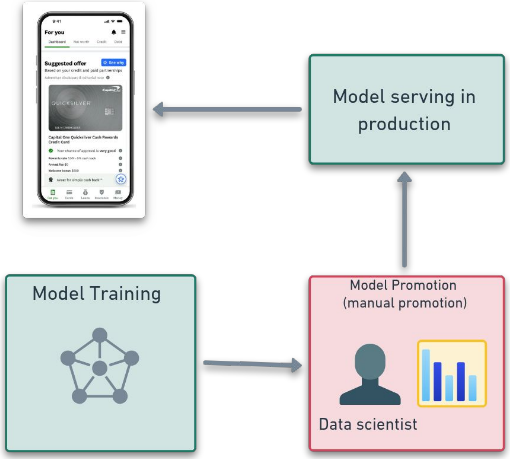
CSI

Decile	Score cutoffs	Baseline- DEV sample		OOT sample		CSI
		Frequency	Percent	Frequency	Percent	
1	830-850	402345	8.5%	395123	9.34%	0.08%
2	810-830	734118	15.5%	700899	16.57%	0.07%
3	780-810	712789	15.0%	612321	14.47%	0.02%
4	730-780	602531	12.7%	501345	11.85%	0.06%
5	700-730	526797	11.1%	413235	9.77%	0.17%
6	660-700	438768	9.3%	337678	7.98%	0.19%
7	530-660	387675	8.2%	336987	7.97%	0.01%
8	460-530	322866	6.8%	322855	7.63%	0.09%
9	400-460	310345	6.5%	309899	7.33%	0.09%
10	300-400	300234	6.3%	300012	7.09%	0.09%
TOTAL		4738468	100.0%	4230354	100.00%	0.87%

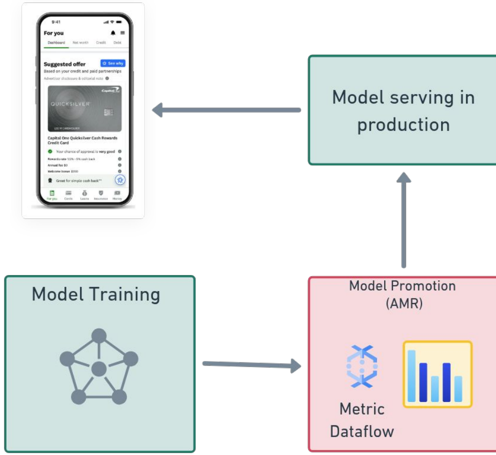
PSI/CSI=Σ(%Actual − %Expected)× ln(%Actual / %Expected)

- Actual %: The percentage of observations in a specific bin for the new or current dataset (production data).
- Expected %: The percentage of observations in the same bin for the original or baseline dataset (e.g., training data).

How Model Monitoring & Dataflow's Role in ML Pipeline Automation

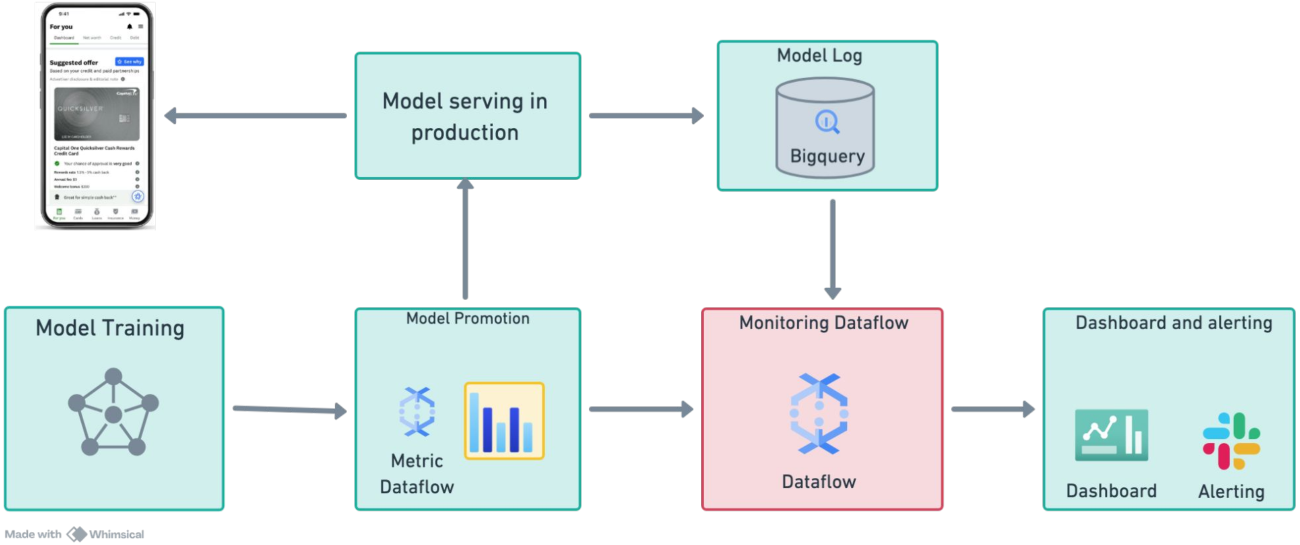


- Dataflow automated post-model preparation and evaluation in our ML pipeline.
- Evaluation metrics (e.g., log loss, precision) determined model promotion to production.
- This integration streamlined ML pipeline automation.
- We aimed to advance beyond fixed retraining schedules.
- Our goal was to implement data-driven drift analysis for smarter retraining decisions.



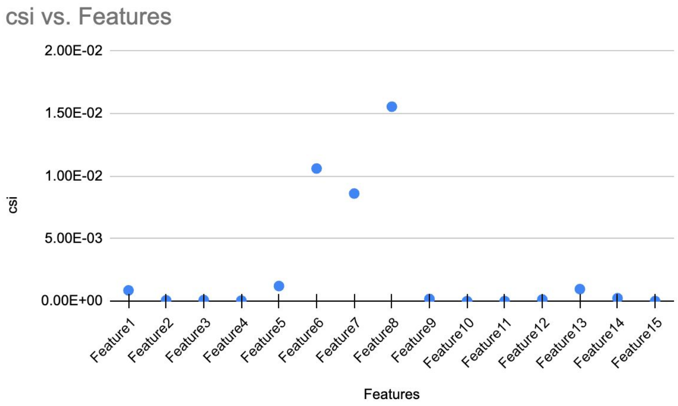
How We Do Model Monitoring? (High Level Design)

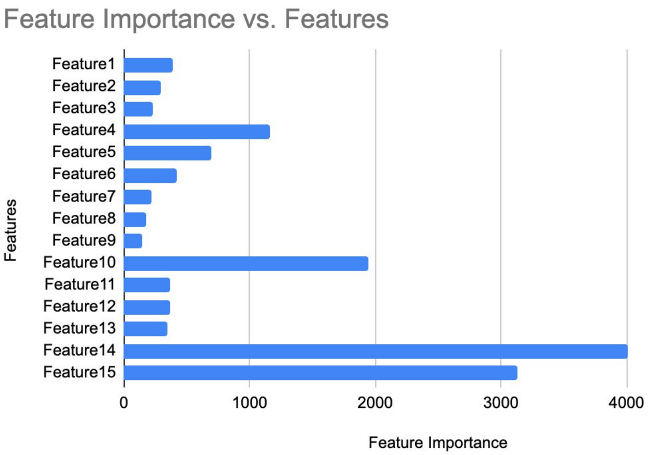
- During model promotion, metrics are calculated using Dataflow to compare control and challenger models.
- If promotion metrics are favorable, decile baseline distributions are established from the promoted model.
- Monitoring Dataflow jobs analyze logs from the promoted model to calculate PSI/CSI.
- These PSI/CSI calculations are then used to alert on detected drift.



Maneuvering Through Drift

Responding to Drift (Monitoring PSI Drift and Remediating)



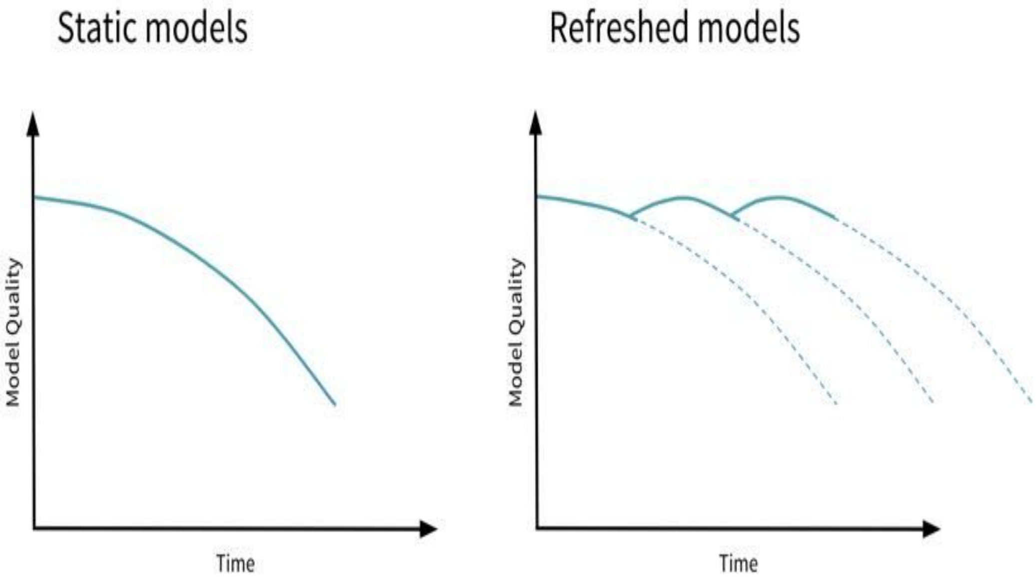


- Monitor PSI drift above the
- Isolate drift using CSI and feature importance
- Pinpoint specific features driving the drift
- Retrain or adjust model to remediate drift

What is the Impact ?

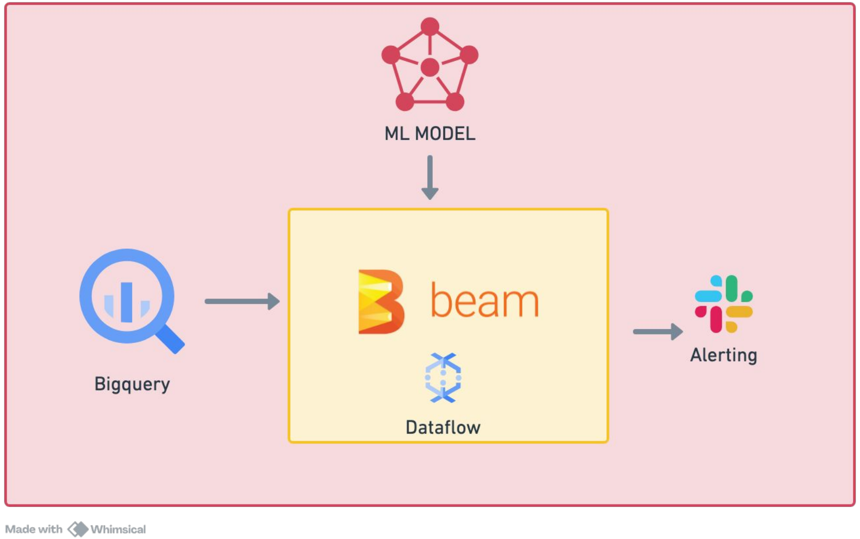
Optimized Resource & Cost Efficiency

- **Retrain** models only when **significant data drift** is detected
- Reduce **unnecessary** computation and **cloud costs**
- **Leverage Dataflow** for **scalable** and **efficient** resource allocation



Transforming Model Management with Dataflow

- **Automate large-scale** monitoring of multiple models
- Streamline oversight and tracking using Dataflow



Credit Karma’s Model Management

Credit Karma employs 100+ ML models to optimize user experience.
Models and features undergo retraining at distinct cadences (e.g., weekly for high-volatility features, quarterly for stable ones).
Proactive alert-driven actions include:

- Adjusting retraining frequency based on drift severity.
- Triggering immediate model updates upon signal degradation detection.
- Re-engineering features if data drift persists.

Key Outcomes:

- Maintained model accuracy despite shifting data landscapes.
- Minimized user-impacting errors through rapid response protocols.

Driving Financial Progress Through Engineering Excellence

In Credit Karma's rapidly changing landscape, delivering value efficiently is paramount.

Apache Beam provides the necessary agility and broad capability support.

This empowers us to continuously innovate and ship rapidly across diverse needs.


Q&A