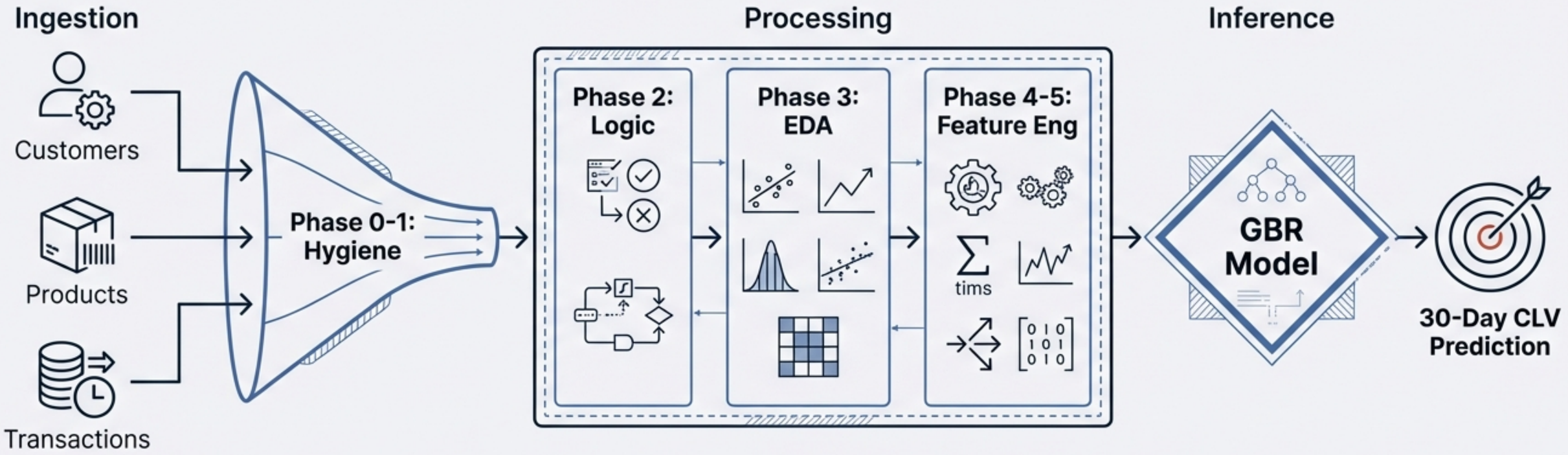
Predictive Customer Lifetime Value (CLV): An End-to-End Engineering Pipeline

From Raw Transaction Logs to Gradient Boosting Predictions



OBJECTIVE:
Predict 30-day future spend

STACK:
Python, Pandas, Scikit-Learn, Joblib

DATASET:
Synthetic Retail (20k Customers, 1k Products)

Prediction quality depends on the integrity of the data pipeline

We are building a system to predict a customer's spend over the next 30 days based on historical behaviour. Success requires a full-stack approach, treating feature engineering as the primary driver of accuracy.

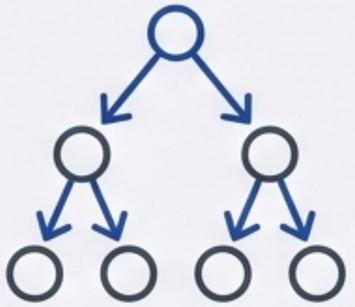
01. Hygiene

Solving **real-world data messiness**. We simulate defects like nulls, duplicates, and "time-travel" paradoxes to force rigorous cleaning protocols.

02. Engineering

Transforming raw logs into behavioural signals. Moving beyond simple averages to capture Momentum, Loyalty, Seasonality, and Category Affinity.

03. Modelling



Deploying a Gradient Boosting Regressor. We use non-linear tree-based models to capture complex interactions between recency and spend.

KEY INSIGHT: The pipeline explicitly prevents 'Look-Ahead Bias'—the most common failure mode in CLV modelling—by strictly segregating training data from future target windows.

Phase 0: Simulating a realistic, high-entropy retail environment

The Entities

Customers: 20,000 unique profiles (Gold/Silver/Bronze status)

Products: 1,000 items (Electronics, Fashion, Grocery)

Period: Jan 2022 – Dec 2025

- **Behaviour:** Weighted Random (25% High Value, 50% Medium, 25% Low)

The Mess / Intentional Defects

```
// INJECTED NOISE & DEFECTS

duplicates = pd.concat([df, df.sample(400)])
   // 400 Duplicate Customers

trans_dupes = pd.concat([df, df.sample(1200)])
   // 1,200 Duplicate Transactions

refunds = amount * -1 // 2% Random Refunds

nulls = np.nan      // Missing Emails & Stock Levels
```

Phase 1: Enforcing strict data typing and intelligent imputation

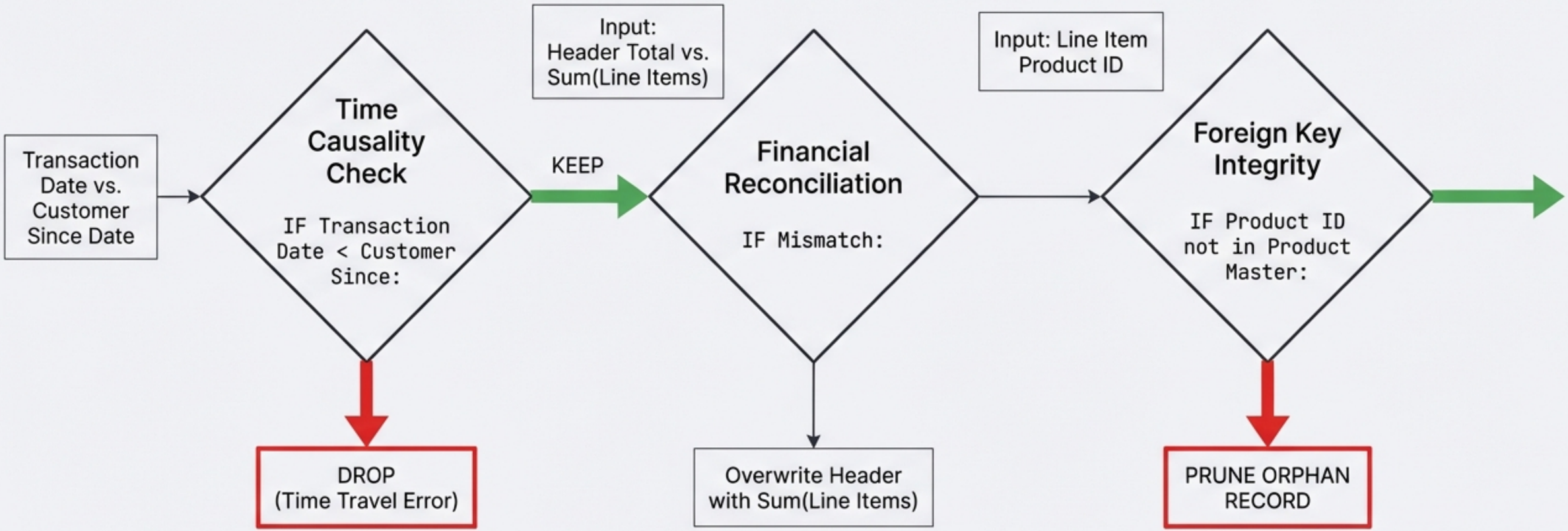
Transformation Table

Raw State		Clean State
transaction_date: Object (String)	pd.to_datetime(errors='coerce') →	**transaction_date:** datetime64[ns]
segment_id: NaN	fillna('Unknown') →	**segment_id:** 'Unknown' (Signal Preserved)
current_stock_level: NaN	fillna(median) →	**current_stock_level:** 142 (Median)

CRITICAL LEAKAGE CHECK

We explicitly do NOT impute ‘last_purchase_date’ during this phase. Imputing this using transaction logs before the train/test split would introduce look-ahead bias.

Phase 2: Validating business logic and relational integrity



Transformation: Applied np.log1p to spend data to handle right-skewed monetary distributions.

Phase 3: Profiling the customer base and transaction coverage

Repeat Customer Rate

Display % (Customers > 1 Tx)
JetBrains Mono

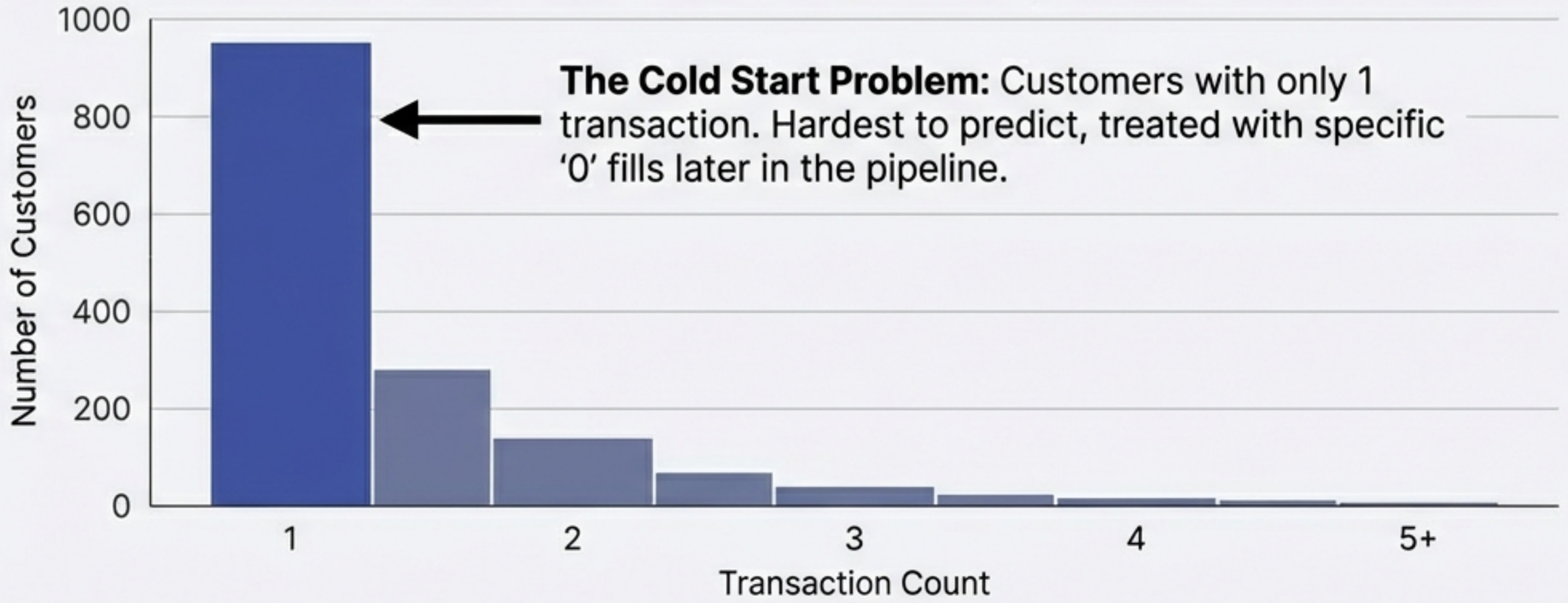
Date Coverage

Jan 2022 — Dec 2025
JetBrains Mono

Active Period

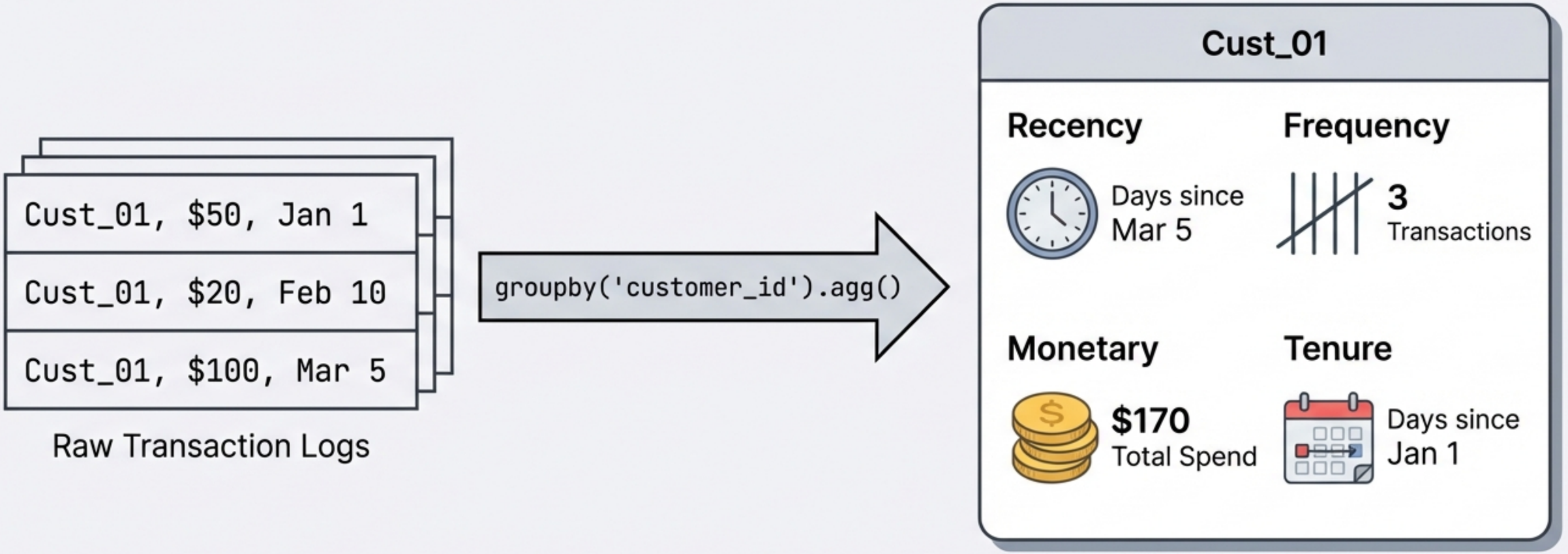
Days between First and Last Purchase
JetBrains Mono

Distribution of Transaction Counts per Customer



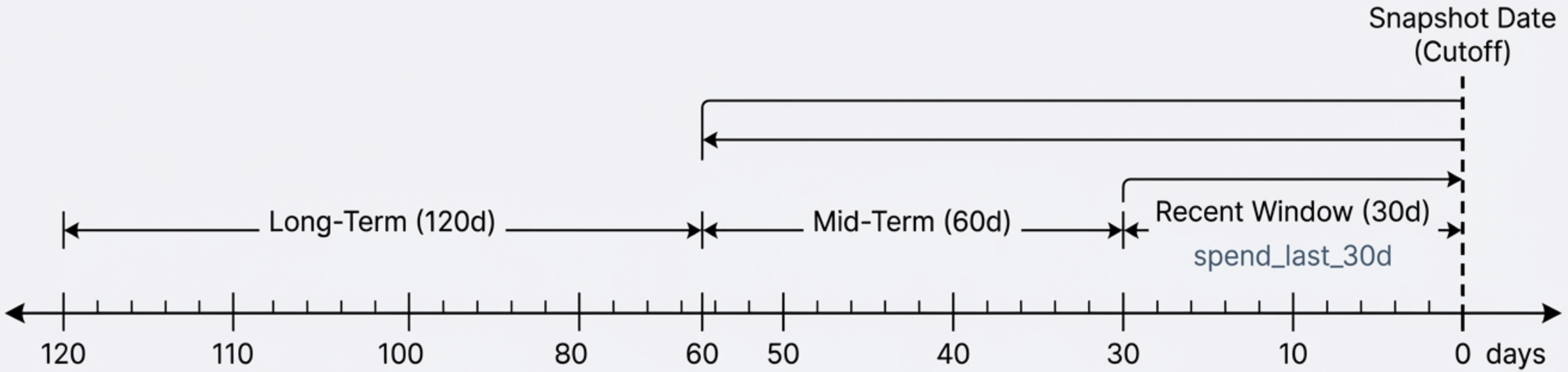
Automated Checks:
Null %, Duplicate Primary Keys, Numeric Summaries.
JetBrains Mono

Feature Engineering I: Transforming logs into behavioural baselines (RFM)



These static aggregates provide the baseline value of a customer but fail to capture the trajectory of their behaviour.

Feature Engineering II: Capturing behavioural drift with time-windowed aggregates




```
def window_agg_safe(df, start_day, end_day):
    start   = cutoff - pd.Timedelta(days=end_day)
    end     = cutoff - pd.Timedelta(days=start_day)
    return df[(date > start) & (date <= end)]
```

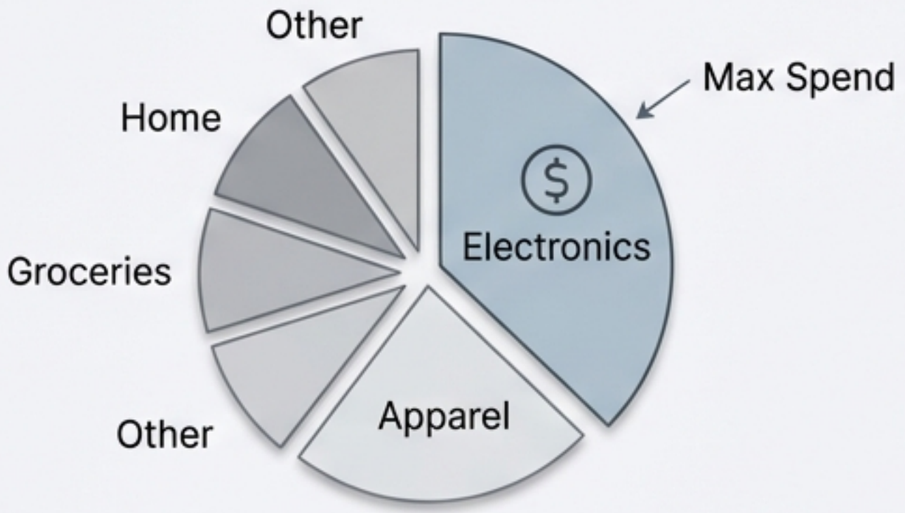
Why? To detect if a customer is “cooling down” (High historic spend vs. Low recent spend) or “heating up”.

Feature Engineering III: Deriving signals from product affinity

Affinity Metrics

Dominant Category

The single category where a customer spends the most money.



Imputation Note: Filled with “Unknown” for zero-purchase customers.

Diversity & Dependency

Category Dependency Ratio

$$\text{Ratio} = \frac{\text{Unique Categories}}{\text{Unique Products}}$$

Specialist (Low Ratio) 1 Category / 5 Products = 0.2

Category: 1 Products: 5 Ratio: 0.2

Generalist (High Ratio) 5 Categories / 5 Products = 1.0

Categories: 5 Products: 5 Ratio: 1.0

Understanding *what* they buy allows the model to segment users (e.g., Tech Enthusiasts vs. Grocery Shoppers).

Feature Engineering IV: Second-order signals (Momentum & Seasonality)

Spend Momentum

```
spend_last_30d /
spend_last_90d + 1
```

Insight: > 1.0 indicates acceleration in spending.

Seasonality (YoY)

```
spend_same_month_last_year
```

Insight: Does the customer always buy in this specific month?

Purchase Regularity

```
tenure_days /
total_transactions
```

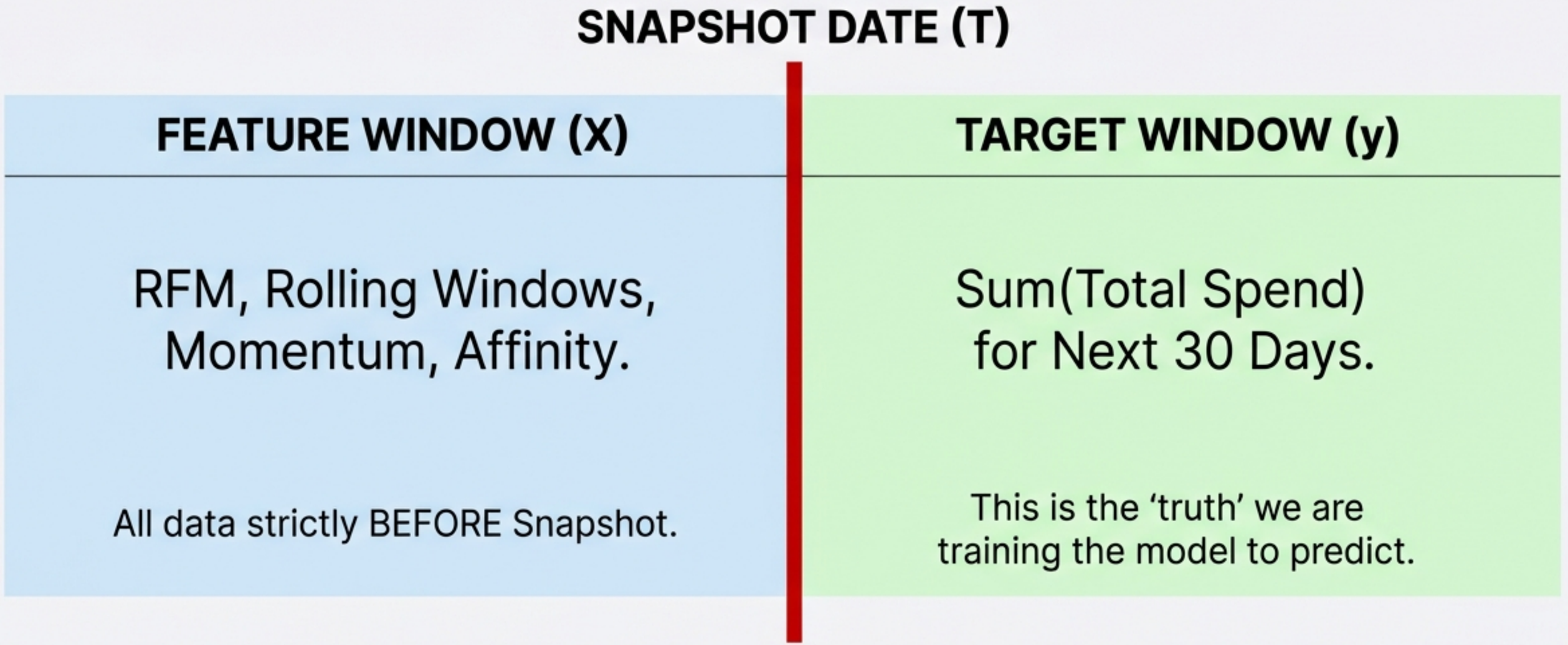
Insight: Average days between purchases (Stability metric).

Interaction Terms

```
total_loyalty_points /
    recency_days
```

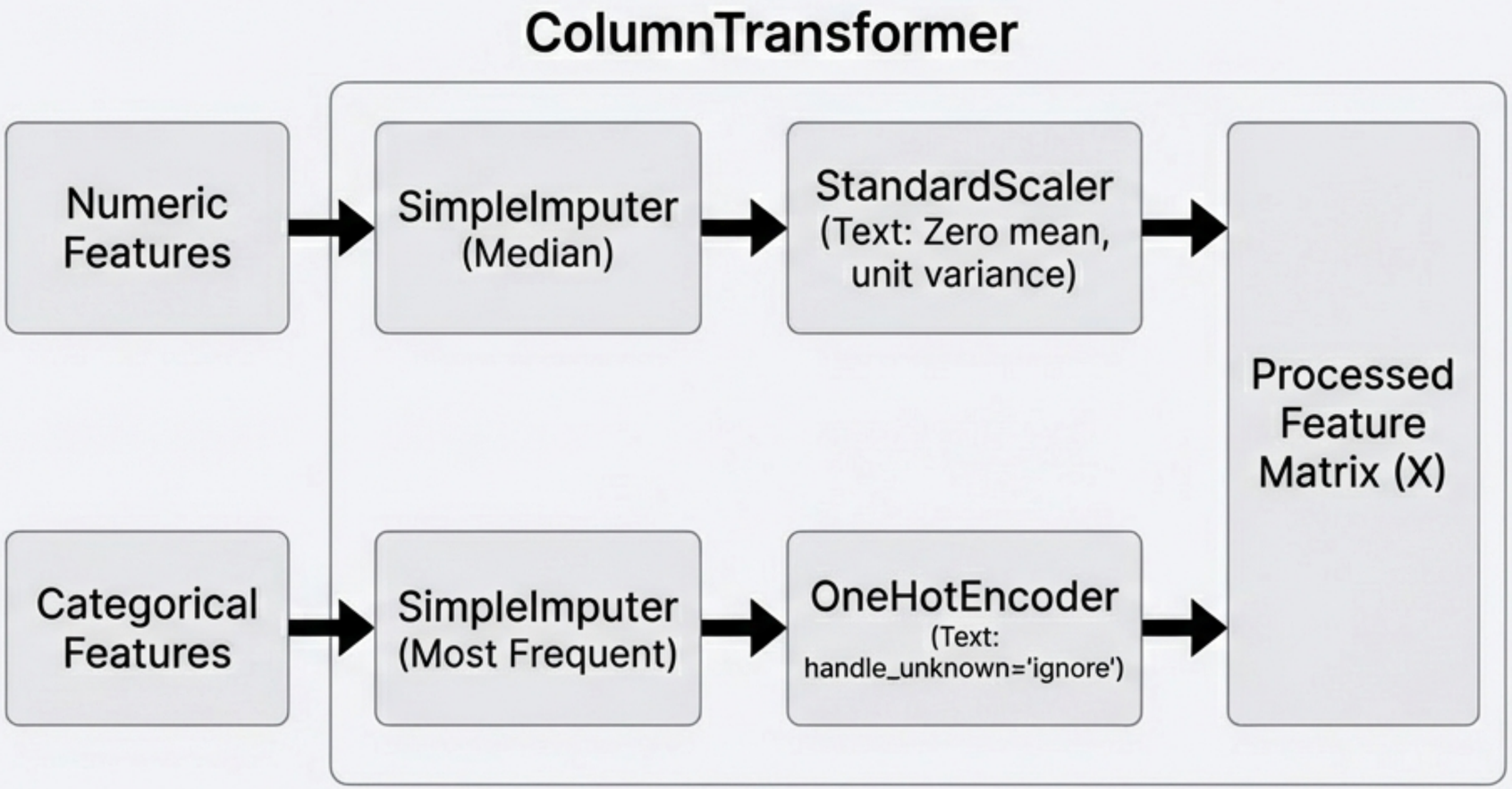
Insight: High points + High Recency days = At-Risk VIP.

Phase 4: Target creation and the 'Snapshot' strategy



Target Handling: Customers with NO purchases in the target window are explicitly filled as 0 (Zero Value), **not dropped.** The model must learn to predict churn.

Phase 5: Building a production-ready preprocessing pipeline



Leakage Removal

- Dropped: `customer_id`
- Dropped: `first_transaction_date`
- Dropped: `last_transaction_date`
- Dropped: `last_transaction_date`

Phase 6: Modelling with Gradient Boosting Regressor

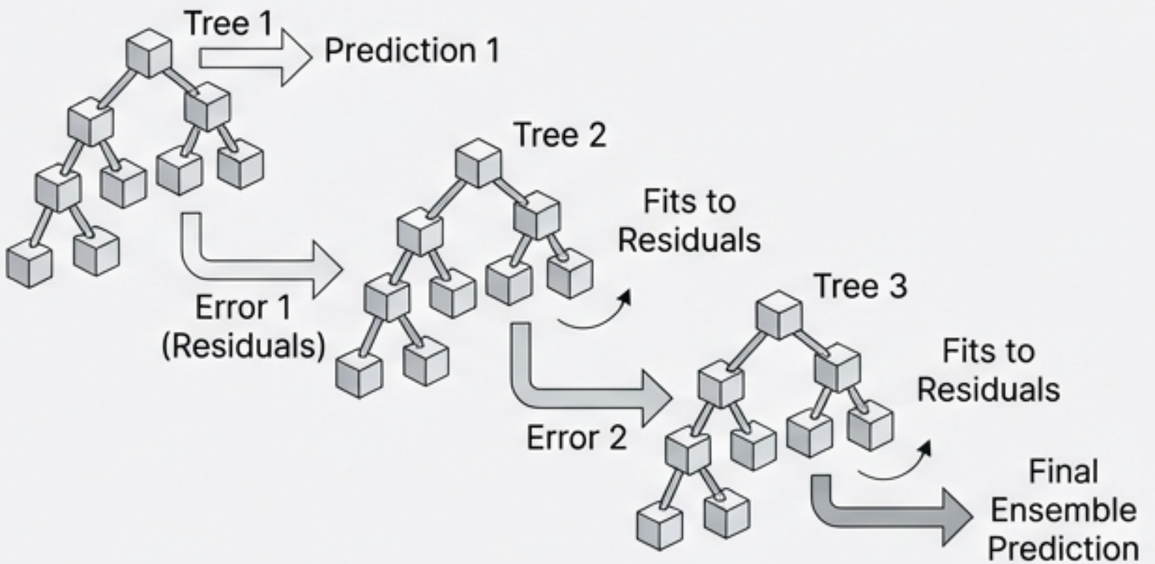
Algorithm: sklearn.ensemble.GradientBoostingRegressor

Why GBR?

1. Handles mixed data types efficiently.
2. Captures non-linear interactions (e.g., Diminishing returns on tenure).
3. Robust to outliers compared to Linear Regression.

Configuration

```
gbr_model = GradientBoostingRegressor(
    n_estimators=300,
    learning_rate=0.05,
    max_depth=4,
    random_state=42
)
```



Evaluation metrics and deployment artifacts

MAE

Mean Absolute Error
Inter Regular

Average dollar error per prediction.

RMSE

Root Mean Squared Error
Inter Regular

Penalizes large outlier misses.

R^2

Coeff. of Determination
Inter Regular

Variance explained by model.

`gradient_boosting_30d_clv_model.pkl`
JetBrains Mono

→

Serialized via `joblib.dump`. Contains full preprocessing pipeline + trained model for immediate API serving.

Technical Editorial

Summary: A robust framework for value prediction

Key Takeaways

Checklist style

- ✓ **Hygiene First:** Logical validation (time-travel checks) prevents silent failures.
- ✓ **Feature Power:** Rolling windows and Momentum signals outperform static averages.
- ✓ **Leakage Proof:** Strict date-based splitting ensures the model learns predictive patterns, not history.

Future Roadmap

Arrow list style

- → **Hyperparameter Tuning:** Implement GridSearch for optimal tree depth.
- → **Deep Learning:** Test LSTM/Transformers for sequential pattern recognition.
- → **External Data:** Integrate holiday calendars and macroeconomic indicators.

End of Pipeline. Inter Regular.