**1. Customer Lifetime Value (LTV) Scoring with RFM and Weighted Normalization**

**Use pandas & numpy to compute normalized RFM metrics and combine with weights.**

SCRIPT\_REAL(

"

import pandas as pd

import numpy as np

recency = pd.Series(\_arg1)

frequency = pd.Series(\_arg2)

monetary = pd.Series(\_arg3)

# Min-max normalization (add epsilon to avoid div by zero)

epsilon = 1e-9

r\_norm = (recency.max() - recency) / (recency.max() - recency.min() + epsilon)

f\_norm = (frequency - frequency.min()) / (frequency.max() - frequency.min() + epsilon)

m\_norm = (monetary - monetary.min()) / (monetary.max() - monetary.min() + epsilon)

# Weighted sum for LTV score

ltv\_score = 0.4 \* f\_norm + 0.4 \* m\_norm + 0.2 \* r\_norm

return ltv\_score.tolist()

",

[Days Since Last Purchase], // calculated in Tableau: TODAY() - MAX([Order Date])

[Order Count], // number of orders per customer

SUM([Sales]) // total sales per customer

)

**2. Churn Prediction Proxy: Rule-based Scoring with Moving Average and Threshold**

**Instead of ML, build a rule-based churn score using numpy moving averages and thresholds**

SCRIPT\_REAL(

"

import numpy as np

sales = np.array(\_arg1)

# Calculate 3-period moving average

mov\_avg = np.convolve(sales, np.ones(3)/3, mode='valid')

# Pad the start to match length

mov\_avg = np.pad(mov\_avg, (len(sales) - len(mov\_avg), 0), 'constant', constant\_values=(mov\_avg[0],))

# Churn risk if recent sales fall below 50% of moving average

churn\_risk = (sales < 0.5 \* mov\_avg).astype(float)

return churn\_risk.tolist()

",

SUM([Sales])

)

**Explanation**

* Checks if recent sales are dropping significantly relative to recent average.
* Returns 1 if likely churn, 0 otherwise.
* Use as color or filter in Tableau.

**3. Linear Trend Slope of Profit Over Time (Using numpy polyfit)**

**Compute the slope of profit trend over time (per category or region)**

SCRIPT\_REAL(

"

import numpy as np

profit = np.array(\_arg1)

n = len(profit)

x = np.arange(n)

if n < 2:

slope = 0

else:

slope = np.polyfit(x, profit, 1)[0]

# Return slope repeated to match input length

return [slope] \* n

",

SUM([Profit])

)

**Usage Tips for Tableau:**

* Set **Compute Using** → Time dimension (e.g., Order Date)
* Partition by Category or Region as needed
* Use slope to identify improving or declining profit trends