Master DataFrame Calculations Documentation

Overview

This document explains all calculated metrics in the Master Product Performance DataFrame used for demand forecasting.

📊 Base Data Sources

- **Sales Data:** `sales_data_df` - Product sales by date
- **Customer Feedback:** `customer_feedback_df` - Reviews with sentiment
- **Product Catalog:** `product_catalog_df` - Product attributes and release dates
- **Search Trends:** `search_trends_df` - Search query patterns

🔢 Calculated Metrics

1. Monthly Sales Aggregation

Purpose: Convert daily sales to monthly time series

Method: Group by product and month, sum quantities

```python
monthly_sales = sales_data_df.groupby(['productId', 'year_month']).agg({
    'quantitySold': 'sum',
    'timestamp': 'count'  # Transaction count
}).reset_index()
```

Output:

- `monthly_sales`: Total units sold per product per month
- `transaction_count`: Number of sales transactions per month

---

2. Month-over-Month (MoM) Growth Rate

Purpose: Measure short-term sales momentum

Formula: `((Current Month – Previous Month) / Previous Month) × 100`

```python
# Sort data by product and time
master_df = master_df.sort_values(['productId', 'year_month'])

# Calculate previous month sales using shift
master_df['prev_month_sales'] = master_df.groupby('productId')['monthl

# Calculate MoM growth rate
master_df['mom_growth_rate'] = (
    (master_df['monthly_sales'] - master_df['prev_month_sales']) /
    (master_df['prev_month_sales'] + 1e-10) * 100
)

# Cap extreme values at ±500%
master_df['mom_growth_rate'] = master_df['mom_growth_rate'].clip(-500,
```

Example:

- Product P1001 January sales: 100 units
- Product P1001 February sales: 120 units
- MoM Growth = (120-100)/100 × 100 = **+20%**

Handling Edge Cases:

- **First month:** No previous data → `NaN`
- **Zero previous sales:** Add small value (1e-10) to avoid division by zero
- **Extreme values:** Capped at ±500% to prevent outliers

---

3. Year-over-Year (YoY) Growth Rate

Purpose: Measure long-term growth trends (removes seasonality)

Formula: `((Current Month - Same Month Previous Year) / Same Month Previous Year) × 100`

```python
# Get sales from 12 months ago
master_df['prev_year_sales'] = master_df.groupby('productId')['monthly

# Calculate YoY growth rate
master_df['yoy_growth_rate'] = (
    (master_df['monthly_sales'] - master_df['prev_year_sales']) /
    (master_df['prev_year_sales'] + 1e-10) * 100
)

# Cap extreme values at ±500%
master_df['yoy_growth_rate'] = master_df['yoy_growth_rate'].clip(-500,
```

Example:

- Product P1001 January 2023: 100 units
- Product P1001 January 2024: 130 units
- YoY Growth = (130-100)/100 × 100 = **+30%**

Handling Edge Cases:

- **Less than 12 months data:** No year-ago data → `NaN`
- **33% missing values:** Expected for YoY calculation

---

4. Time-Weighted Sentiment Analysis

Purpose: Give recent customer feedback more importance

Method: Apply recency weights based on review age

python

```python
# Calculate review age in months
max_date = feedback_clean['date'].max()
feedback_clean['months_old'] = ((max_date - feedback_clean['date']).dt

# Define time-based weights
def get_recency_weight(months_old):
    if months_old <= 3:    # Recent (0-3 months)
        return 1.0
    elif months_old <= 6:  # Mid-term (3-6 months)
        return 0.7
    else:                  # Older (6+ months)
        return 0.4

# Apply weights
feedback_clean['recency_weight'] = feedback_clean['months_old'].apply(
feedback_clean['weighted_sentiment'] = (feedback_clean['vader_sentimen
                                       feedback_clean['recency_weight'

# Monthly aggregation
monthly_sentiment = feedback_clean.groupby(['productId', 'year_month']
    'weighted_sentiment': 'sum',
    'recency_weight': 'sum',
    'vader_sentiment_cleaned': ['count', 'mean'],
    'rating': 'mean'
}).reset_index()

# Calculate final weighted average
monthly_sentiment['avg_sentiment_weighted'] = (
    monthly_sentiment['weighted_sentiment_sum'] /
    (monthly_sentiment['total_weights'] + 1e-10)
)
```

Weight Distribution:

Review Age	Weight	Rationale
0-3 months	1.0	Most relevant for current trends
3-6 months	0.7	Somewhat relevant
6+ months	0.4	Less relevant but still useful

---

5. Seasonal Mapping

Purpose: Categorize months into US fashion seasons

Method: Map calendar months to business seasons

```python
def get_season(month):
    if month in [3, 4, 5]:      # March, April, May
        return 'Spring'
    elif month in [6, 7, 8]:    # June, July, August
        return 'Summer'
    elif month in [9, 10, 11]:  # September, October, November
        return 'Fall'
    else:                       # December, January, February
        return 'Winter'

# Apply to all date columns
sales_data_df['season'] = sales_data_df['timestamp'].dt.month.apply(ge
master_df['season'] = master_df['year_month'].dt.month.apply(get_seaso
```

Rationale: Based on US fashion industry seasonal patterns and weather

---

6. Days Since Release

Purpose: Measure product lifecycle stage

Method: Calculate days between month timestamp and release date

```python
# Convert year_month to timestamp for calculation
master_df['month_timestamp'] = master_df['year_month'].dt.to_timestamp

# Calculate days since release
master_df['days_since_release'] = (
    master_df['month_timestamp'] - master_df['releaseDate']
).dt.days
```

Interpretation:

- **Positive values:** Product has been released
- **Negative values:** Pre-release period (filtered out in our analysis)
- **0:** Launch month

---

7. New Product Flag

Purpose: Identify recently launched products

Method: Flag products released within last 6 months

```python
master_df['is_new_product'] = master_df['days_since_release'] <= 180
```

Business Logic: New products need different forecasting approaches

---

8. Moving Averages (3-Month)

Purpose: Smooth out short-term fluctuations

Method: Rolling window calculations

```python
# Sales 3-month moving average
master_df['sales_3month_ma'] = (
    master_df.groupby('productId')['monthly_sales']
    .rolling(window=3, min_periods=1)
    .mean()
    .values
)

# Sentiment 3-month moving average
master_df['sentiment_3month_ma'] = (
    master_df.groupby('productId')['avg_sentiment_weighted']
    .rolling(window=3, min_periods=1)
    .mean()
    .values
)
```

Parameters:

- **Window:** 3 months
- **Min periods:** 1 (allows calculation even with limited data)

---

9. Positive Review Percentage

Purpose: Measure customer satisfaction ratio

Method: Calculate percentage of positive sentiment reviews

```python
# In monthly sentiment aggregation
monthly_sentiment['positive_reviews_pct'] = (
    monthly_sentiment['positive_reviews'] /
    monthly_sentiment['review_count'] * 100
)
```

Definition: Percentage of reviews with positive sentiment (VADER > 0.05)

🧹 Data Quality Handling

Missing Value Strategy

Column	Missing Reason	Handling
`mom_growth_rate`	First month per product	Keep as `NaN`
`yoy_growth_rate`	<12 months data	Keep as `NaN`
`avg_sentiment_weighted`	No reviews	Fill with 0.0
`review_count`	No reviews	Fill with 0
`positive_reviews_pct`	No reviews	Fill with 50% (neutral)

Outlier Treatment

- **Growth rates:** Capped at ±500% to handle zero-sales scenarios
- **Sentiment scores:** Bounded to [-1, +1] (natural VADER range)
- **Sales values:** No capping (all values are realistic)

Pre-Release Data Filtering

- **Issue:** 50.9% of records had sales before product release
- **Solution:** Filtered out all records where `days_since_release < 0`
- **Impact:** Retained 9,716/10,000 products with realistic sales patterns

📊 Final Master DataFrame Schema

Column	Type	Description	Range/Values
`productId`	object	Unique product identifier	P1001-P10000
`year_month`	period	Month of sales data	2022-05 to 2025-04
`monthly_sales`	int	Units sold in month	0-164
`season`	object	Business season	Spring/Summer/Fall/Winter
`transaction_count`	int	Number of sales transactions	0-8
`mom_growth_rate`	float	Month-over-month growth %	-500 to +500
`yoy_growth_rate`	float	Year-over-year growth %	-500 to +500
`category`	object	Product category	30 unique categories
`modifiers`	object	Product attributes	Text (comma-separated)
`keywords`	object	SEO keywords	Text (comma-separated)
`releaseDate`	datetime	Product launch date	2022-05-01 to 2025-04-30
`release_season`	object	Season of product launch	Spring/Summer/Fall/Winter
`avg_sentiment_weighted`	float	Time-weighted sentiment	-0.895 to +0.961
`review_count`	int	Number of reviews in month	0-8
`positive_reviews_pct`	float	% of positive reviews	0-100
`avg_rating`	float	Average star rating	1.0-5.0

Column	Type	Description	Range/Values
`days_since_release`	int	Days since product launch	0-1066
`is_new_product`	bool	Released within 6 months	True/False
`sales_3month_ma`	float	3-month sales moving average	0-113
`sentiment_3month_ma`	float	3-month sentiment moving average	-0.895 to +0.961

Total Records: 176,693 product-month combinations

Date Range: May 2022 - April 2025 (36 months)

Products: 9,716 with valid post-release data