Preprocessing to Model Selection Rationale

Sales Prediction for Pharmaceutical Distribution Companies by Time-Series Analysis

> **Document Purpose:** This document explains the relationship between data preprocessing choices and the selection of three forecasting models (SARIMA, Prophet, SVR) for pharmaceutical sales prediction.

---

1. Executive Summary

This project implements a **comprehensive multi-model comparison** approach where the same preprocessing steps prepare data for three fundamentally different forecasting algorithms. The preprocessing pipeline is specifically designed to:

1. **Enable statistical model requirements** (stationarity for SARIMA)
2. **Support automated seasonality detection** (for Prophet)
3. **Prepare supervised learning format** (for SVR)

The key insight is that preprocessing decisions—particularly **weekly resampling**, **stationarity transformations**, and **temporal feature engineering**—directly enable the comparison of statistical, additive regression, and machine learning approaches on the same pharmaceutical sales data.

---

2. Data Characteristics Driving Preprocessing Decisions

2.1 Multi-Granularity Source Data

File	Frequency	Records	Primary Use
`saleshourly.csv`	Hourly	~52,560	Intraday pattern analysis
`salesdaily.csv`	Daily	~2,190	**Primary data source**
`salesweekly.csv`	Weekly	302	Benchmark for resampling
`salesmonthly.csv`	Monthly	~72	Long-term seasonality

Preprocessing Decision: Use `salesdaily.csv` as primary source and resample to weekly.

Rationale:

- Daily data is too noisy for accurate long-term forecasting
- Weekly aggregation smooths day-of-week effects while preserving seasonal patterns
- Provides sufficient data points (~300 weeks) for model training

2.2 Drug Category Structure

Eight ATC-classified pharmaceutical categories requiring independent forecasting:

Code	Type	Pattern Characteristics
M01AB	Anti-inflammatory	Moderate seasonality
M01AE	Anti-inflammatory	Increasing trend
N02BA	Analgesics (Aspirin)	Weak seasonality
N02BE	Analgesics (Paracetamol)	Strong seasonality
N05B	Anxiolytics	High variance/noise
N05C	Hypnotics/Sedatives	High variance/noise
R03	Respiratory	Strong annual seasonality
R06	Antihistamines	Strong annual seasonality

Preprocessing Decision: Apply same preprocessing pipeline to each category independently.

Model Implication: Different categories may perform better with different models based on their pattern characteristics.

---

3. Preprocessing Pipeline Breakdown

3.1 Step 1: Data Loading and Column Cleanup

```
df_daily = pd.read_csv("salesdaily.csv")
df_daily.drop(['Year', 'Month', 'Hour', 'Weekday Name'], axis=1, inplace=True)
df_daily['datum'] = pd.to_datetime(df_daily['datum'])
df_daily.rename(columns={'datum': 'Date'}, inplace=True)
df_daily = df_daily.set_index('Date')
```

Why Drop Metadata Columns?

Dropped Column	Reason
`Year`	Redundant - extractable from datetime index
`Month`	Redundant - extractable from datetime index
`Hour`	Always same value for daily data
`Weekday Name`	Redundant - extractable from datetime index

Model Implications:

- **SARIMA:** Requires clean datetime index for time-series operations
- **Prophet:** Will auto-extract temporal features from `ds` column
- **SVR:** Removes potential feature leakage from pre-computed columns

3.2 Step 2: Weekly Resampling

```
df = df_daily['2014-01-02':'2019-10-08'].resample('W').sum()
```

Why Weekly Aggregation?

Factor	Daily Data	Weekly Data
Noise Level	High	Reduced
Data Points	~2,190	~302
Weekly Effects	Visible	Smoothed out
Seasonality	Harder to detect	Clearer patterns
Model Complexity	Higher	More manageable

Model-Specific Benefits:

Model	Benefit of Weekly Data
SARIMA	Cleaner ACF/PACF patterns for parameter selection
Prophet	More stable trend estimation
SVR	Fewer timesteps needed in sliding window

3.3 Step 3: Stationarity Testing

```
# Rolling statistics visualization
rolmean = pd.Series(y).rolling(window=12).mean()
rolstd = pd.Series(y).rolling(window=12).std()

# Augmented Dickey-Fuller Test
dftest = adfuller(timeseries.dropna(), autolag='AIC')

# KPSS Test
statistic, p_value, n_lags, critical_values = kpss(series, regression="ct",
nlags="auto")
```

Why Two Stationarity Tests?

Test	Null Hypothesis	P < 0.05 Means	Purpose
ADF	Unit root exists (non-stationary)	**Stationary**	Detect trend
KPSS	Series is stationary	**Non-stationary**	Confirm trend

Decision Matrix:

ADF Result	KPSS Result	Conclusion	Action
Stationary	Stationary	**Confirmed stationary**	Use d=0 in ARIMA
Non-stationary	Non-stationary	**Confirmed non-stationary**	Apply differencing
Mixed results	Mixed results	**Ambiguous**	Try both approaches

Model Implications:

- **SARIMA:** Directly uses stationarity information for d parameter
- **Prophet:** Handles non-stationarity automatically (no action needed)
- **SVR:** Scaling handles non-stationarity implicitly

3.4 Step 4: Stationarity Transformations

4.4.1 Detrending (Z-Score Normalization)

```
y_detrend = (y - y.rolling(window=12).mean()) / y.rolling(window=12).std()
```

Mathematical Formulation: $$y_{detrend} = \frac{y_t - \mu_{rolling,t}}{\sigma_{rolling,t}}$$

What This Achieves:

- Removes trend by subtracting rolling mean
- Standardizes variance by dividing by rolling standard deviation
- Creates a series with approximately zero mean and unit variance

4.4.2 First-Order Differencing

```
first_diff = df['M01AB'].diff()
```

Mathematical Formulation: $$y'_t = y_t - y_{t-1}$$

Why Differencing for SARIMA?

Original Data	Differenced Data
May have trend	Trend removed
Non-constant mean	Approximately zero mean
Correlated observations	More independent

The `d` Parameter:

- d=0: Already stationary (no differencing)
- d=1: First differencing (most common)
- d=2: Second differencing (rare, for very strong trends)

3.5 Step 5: Autocorrelation Analysis

```
plot_acf(df['M01AB'], lags=30)
plot_pacf(df['M01AB'], lags=30)
```

How ACF/PACF Guide SARIMA Parameters:

Plot	Pattern	Parameter Indication
ACF	Sharp cutoff at lag k	q = k (MA order)
ACF	Slow decay	AR process present
PACF	Sharp cutoff at lag k	p = k (AR order)
PACF	Slow decay	MA process present
Both	Spikes at seasonal lags (12, 24...)	Seasonal component needed

Why This Matters:

- ACF/PACF patterns directly inform (p, q) and (P, Q) parameter selection
- Reduces grid search space significantly
- Provides theoretical basis for model specification

---

4. Model Selection Rationale

4.1 Why SARIMA?

Preprocessing That Enables SARIMA:

Preprocessing Step	SARIMA Requirement Met
Weekly resampling	Regular time intervals
Datetime index	Time-series operations
Stationarity testing	Informs d parameter
Differencing	Achieves stationarity
ACF/PACF analysis	Guides p, q selection

SARIMA Selection Criteria:

```
order = (2, 1, 3)              # (p, d, q)
seasonal_order = (2, 1, 3, 12)  # (P, D, Q, m)
```

Parameter	Value	Justification
p=2	AR(2)	PACF shows 2 significant lags
d=1	First difference	ADF test indicates non-stationarity
q=3	MA(3)	ACF shows 3 significant lags
P=2	Seasonal AR(2)	Seasonal PACF pattern
D=1	Seasonal difference	Annual trend present
Q=3	Seasonal MA(3)	Seasonal ACF pattern
m=12	Monthly seasonality	Annual cycle in weekly data

Grid Search for Optimal Parameters:

```
p = d = q = range(0, 2)
pdq = list(itertools.product(p, d, q))  # 8 combinations
seasonal_pdq = [(x[0], x[1], x[2], 12) for x in pdq]  # 8 seasonal
combinations

# Total: 8 × 8 = 64 parameter combinations tested
# Selection criterion: Minimum AIC
```

Why AIC for Model Selection? $$AIC = 2k - 2\ln(\hat{L})$$

- Balances model fit (likelihood) with complexity (k parameters)
- Penalizes overfitting
- Enables fair comparison across different (p,d,q) specifications

4.2 Why Facebook Prophet?

Preprocessing That Enables Prophet:

Preprocessing Step	Prophet Requirement Met
Datetime column	Required `ds` column
Single target column	Required `y` column
Daily/Weekly data	Handles multiple granularities
No stationarity needed	Handles trends automatically

Prophet Data Preparation:

```
# Prophet requires specific column names
df_daily.columns = ['ds', 'y']  # Rename to Prophet format
```

Why Prophet is Suitable:

Data Characteristic	Prophet Strength
Annual seasonality	Built-in yearly seasonality
Missing data	Robust interpolation
Trend changes	Automatic changepoint detection
Human-interpretable	Component decomposition

Prophet Configuration:

```
m = Prophet(
    interval_width=0.95,      # 95% confidence intervals
    yearly_seasonality=True   # Enable annual patterns
)
future = m.make_future_dataframe(periods=25, freq='M')
```

Configuration	Purpose
`interval_width=0.95`	Uncertainty quantification
`yearly_seasonality=True`	Pharmaceutical annual patterns
`freq='M'`	Monthly forecast granularity

Why Prophet Complements SARIMA:

- No stationarity assumption required
- Automatic handling of multiple seasonalities
- Better for data with irregular patterns
- Provides interpretable components (trend + seasonality)

4.3 Why Support Vector Regression (SVR)?

Preprocessing That Enables SVR:

Preprocessing Step	SVR Requirement Met
Weekly resampling	Regular intervals for sliding window

Preprocessing Step	SVR Requirement Met
MinMaxScaler	Required for distance-based algorithms
Timesteps creation	Supervised learning format

Critical SVR Preprocessing:

Feature Scaling

```
scaler = MinMaxScaler()
train['M01AB'] = scaler.fit_transform(train)
test['M01AB'] = scaler.transform(test)  # Note: Should use transform, not
fit_transform
```

Why MinMaxScaler? $$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Aspect	Without Scaling	With Scaling
Feature range	Varies by drug category	[0, 1] uniform
Distance calculations	Biased by magnitude	Fair comparison
Kernel computation	May overflow	Numerically stable
Convergence	Slow or fails	Fast and reliable

Sliding Window (Timesteps) Creation

```
timesteps = 5
train_data_timesteps = np.array([
    [j for j in train_data[i:i+timesteps]]
    for i in range(0, len(train_data)-timesteps+1)
])[:,:,0]

# Creates: X = [t-4, t-3, t-2, t-1, t], y = t+1
```

Why 5 Timesteps?

Timesteps	Pros	Cons
3	Simple, fast	May miss patterns
5	**Captures weekly effects**	**Good balance**
7	Full week context	More complexity

Timesteps	Pros	Cons
12	Monthly patterns	Overfitting risk

SVR Hyperparameter Selection:

```
param_grid = {
    'kernel': ['rbf'],               # Radial Basis Function
    'gamma': [0.1, 0.01, 0.001], # Influence radius
    'C': [0.1, 1, 10],               # Regularization
    'epsilon': [0.05, 0.1]       # Error tolerance
}
```

Parameter	Optimal	Role
kernel='rbf'	Selected	Non-linear relationships
gamma=0.01	Grid search	Controls smoothness
C=1	Grid search	Bias-variance trade-off
epsilon=0.1	Grid search	ε-insensitive tube width

Why SVR for This Data:

- Captures non-linear relationships between lags
- Handles the noise in pharmaceutical data
- GridSearchCV finds optimal hyperparameters
- Complements statistical methods with ML approach

---

5. Preprocessing-Model Synergy Analysis

5.1 How Preprocessing Enables Multi-Model Comparison

```
┌──────────────────────────────────────────────────────────────────────┐
│                      RAW DATA (salesdaily.csv)                       │
│                      ~2,190 daily observations                       │
└──────────────────────────────────────────────────────────────────────┘
                                   │
                                   ▼
┌──────────────────────────────────────────────────────────────────────┐
│                    COMMON PREPROCESSING                               │
│   • Drop metadata columns                                            │
│   • Convert to datetime index                                        │
│   • Resample to weekly                                               │
│   • Train-test split (chronological)                                 │
└──────────────────────────────────────────────────────────────────────┘
```

```
                                 │
              ┌──────────────────┼──────────────────┐
              ▼                  ▼                  ▼
     ┌─────────────────┐ ┌─────────────────┐ ┌─────────────────┐
     │  SARIMA PATH    │ │  PROPHET PATH   │ │   SVR PATH      │
     ├─────────────────┤ ├─────────────────┤ ├─────────────────┤
     │ + Stationarity  │ │ + Rename to     │ │ + MinMaxScaler  │
     │   testing       │ │   ds, y columns │ │ + Create        │
     │ + ACF/PACF      │ │ + (No transform │ │   timesteps     │
     │   analysis      │ │   needed)       │ │ + 2D tensor     │
     │ + Differencing  │ │                 │ │   format        │
     │   if needed     │ │                 │ │                 │
     └─────────────────┘ └─────────────────┘ └─────────────────┘
              │                  │                  │
              ▼                  ▼                  ▼
     ┌─────────────────┐ ┌─────────────────┐ ┌─────────────────┐
     │ SARIMA(p,d,q)   │ │ Prophet with    │ │ SVR with        │
     │ ×(P,D,Q,m)      │ │ yearly          │ │ GridSearchCV    │
     │ via Grid Search │ │ seasonality     │ │ (RBF kernel)    │
     └─────────────────┘ └─────────────────┘ └─────────────────┘
              │                  │                  │
              └──────────────────┼──────────────────┘
                                 ▼
     ┌───────────────────────────────────────────────────────────────┐
     │                     FAIR RMSE COMPARISON                      │
     │   All models evaluated on same test period                    │
     │   All predictions inverse-transformed to original scale       │
     └───────────────────────────────────────────────────────────────┘
```

5.2 Why This Three-Model Approach?

Aspect	SARIMA	Prophet	SVR
Type	Statistical	Additive Regression	Machine Learning
Assumption	Stationarity	Trend + Seasonality	Feature independence
Seasonality	Explicit (P,D,Q,m)	Automatic	Implicit (via lags)
Interpretability	High	Medium	Low
Best For	Clear ARIMA patterns	Multiple seasonalities	Non-linear relations
Preprocessing	Most extensive	Minimal	Moderate

5.3 Preprocessing Effort vs. Model Complexity Trade-off

```
    Preprocessing Complexity
              ▲
              │
      SARIMA  │ ████████████████████  (Stationarity + ACF/PACF + Differencing)
```

```
               │
         SVR   │ ████████████████          (Scaling + Timesteps)
               │
     Prophet   │ ████████                  (Column renaming only)
               │
               └──────────────────────────────────────► Model Automation
```

Key Insight: More automated models (Prophet) require less preprocessing, while statistical models (SARIMA) require rigorous preprocessing to meet assumptions.

---

6. Category-Specific Model Performance Implications

Based on preprocessing findings, expected model performance varies by drug category:

Category	Pattern Type	Expected Best Model	Reasoning
R03, R06	Strong annual seasonality	SARIMA/Prophet	Clear seasonal patterns favor dedicated seasonality modeling
N02BE	Predictable patterns	Prophet	Automatic trend + seasonality detection
M01AB, M01AE	Increasing trend	SARIMA	Trend-aware differencing helps
N05B, N05C	High noise/variance	SVR	ML handles noisy data better

---

7. Key Preprocessing-Model Dependencies

7.1 Critical Dependencies

If Preprocessing Shows...	Then Model Selection Should...
ADF test: Non-stationary	SARIMA needs d ≥ 1
ACF: Seasonal spikes at 12	SARIMA needs m=12
High noise in residuals	Consider SVR or increase Prophet changepoints
Clear annual pattern	Enable Prophet yearly_seasonality
Strong autocorrelation	SARIMA likely to perform well

7.2 Preprocessing Validates Model Assumptions

Model	Assumption	Preprocessing That Validates
SARIMA	Data is (or can be made) stationary	ADF/KPSS tests

Model	Assumption	Preprocessing That Validates
SARIMA	Autocorrelation structure exists	ACF/PACF analysis
Prophet	Trend and seasonality are additive	Decomposition analysis
SVR	Features are on similar scales	MinMaxScaler applied
SVR	Recent history predicts future	Sliding window validation

---

8. Conclusion

8.1 Preprocessing-Model Selection Summary

The preprocessing pipeline in this project serves three critical purposes:

1. **Enables Fair Comparison:** By creating a common weekly-aggregated dataset, all three models can be compared on equal footing using RMSE.

2. **Meets Model Requirements:**

   - SARIMA: Stationarity achieved through differencing
   - Prophet: Data formatted with `ds` and `y` columns
   - SVR: Features scaled and restructured as supervised learning problem

3. **Informs Parameter Selection:**

   - ACF/PACF → SARIMA (p, q) orders
   - Stationarity tests → SARIMA d parameter
   - Grid search → SVR hyperparameters

8.2 Why This Combination Works

```
┌─────────────────────────────────────────────────────────────────────┐
│                      PHARMACEUTICAL SALES DATA                      │
│                                                                     │
│  Characteristics:                                                   │
│  ✓ Annual seasonality (illness patterns, prescribing cycles)       │
│  ✓ Trend (market growth/decline)                                   │
│  ✓ Noise (daily variation, supply chain)                           │
│  ✓ Multiple independent categories                                 │
└─────────────────────────────────────────────────────────────────────┘
                                │
                                ▼
┌─────────────────────────────────────────────────────────────────────┐
│                      MODEL SELECTION RATIONALE                      │
│                                                                     │
│  SARIMA:  For categories with clear ARIMA structure               │
│  Prophet: For categories needing automatic seasonality detection  │
│  SVR:     For categories with non-linear patterns or high noise   │
│                                                                     │
```

```
│  Result:  Robust forecasting through model diversity                    │
└──────────────────────────────────────────────────────────────────────────┘
```

8.3 Final Recommendations

Recommendation	Rationale
Use ensemble of all three models	Different models excel on different categories
Weekly aggregation is optimal	Balances noise reduction with pattern preservation
Always test stationarity first	Guides SARIMA specification and identifies trends
Scale data for SVR	Critical for kernel-based algorithms
Use chronological train-test split	Prevents data leakage in time series

---

This document explains how preprocessing decisions directly inform and enable the multi-model forecasting approach used in this pharmaceutical sales prediction project.