💊 Sales Prediction for Pharmaceutical Distribution Companies

> **A comprehensive time-series analysis capstone project comparing SARIMA, Support Vector Regression (SVR), and Facebook Prophet for pharmaceutical sales forecasting across 8 drug categories.**

python 3.9 | statsmodels 0.13+ | Prophet 1.0+ | scikit-learn 1.0+

🎯 Project Overview

This capstone project implements and compares **three distinct time-series forecasting models** for predicting pharmaceutical sales at distribution companies:

Model	Type	Strengths
SARIMA	Statistical	Captures seasonality, interpretable parameters
Prophet	Additive Regression	Handles holidays, robust to missing data
SVR	Machine Learning	Non-linear relationships, hyperparameter tuning

Key Objectives

- 📊 **Exploratory Data Analysis** — Seasonality, stationarity, and autocorrelation analysis
- 🔮 **Multi-Model Forecasting** — Independent models per drug category
- 📈 **Performance Comparison** — RMSE benchmarking across all approaches
- 💹 **Business Insights** — Actionable recommendations for inventory planning

---

📁 Repository Structure

```
Sales-Prediction-for-Pharmaceutical-Distribution-Companies-by-Time-Series-
Analysis-main/
│
├── 📓 EDA of the dataset.ipynb                   # Exploratory Data Analysis
(35 cells)
├── 📓 Capstone_Project_Routine_Chemistry.ipynb    # Additional analysis (25
cells)
│
├── 📓 Applying TS models on M01AB.ipynb           # Full pipeline for M01AB
(115 cells)
├── 📓 Applying TS models on M01AE.ipynb           # Full pipeline for M01AE
├── 📓 Applying TS models on N02BA.ipynb           # Full pipeline for N02BA
├── 📓 Applying TS models on N02BE.ipynb           # Full pipeline for N02BE
├── 📓 Applying TS models on N05B.ipynb            # Full pipeline for N05B
├── 📓 Applying TS models on N05C.ipynb            # Full pipeline for N05C
```

```
    ├── 📓 Applying TS models on R03.ipynb          # Full pipeline for R03
    ├── 📓 Applying TS models on R06.ipynb          # Full pipeline for R06
    │
    ├── 📊 salesdaily.csv                            # Daily sales data
    ├── 📊 saleshourly.csv                           # Hourly sales data
    ├── 📊 salesweekly.csv                           # Weekly sales data
    ├── 📊 salesmonthly.csv                          # Monthly sales data
    │
    ├── 📄 README.md                                 # Original project readme
    └── 📖 my_readme.md                              # This comprehensive guide
```

Notebook Organization

Notebook	Purpose	Key Outputs
EDA of the dataset	Data exploration, profiling, visualization	Seasonality patterns, boxplots, trend analysis
Applying TS models on [CATEGORY]	Complete forecasting pipeline	SARIMA, Prophet, SVR predictions per category
Capstone_Project_Routine_Chemistry	Summary analysis	Consolidated findings

📊 Dataset Description

Source

Kaggle Dataset by Milan Zdravković — Pharmaceutical sales data from a single pharmacy Point-of-Sale system.

Data Granularity

File	Rows	Frequency	Primary Use
`saleshourly.csv`	~52,560	Hourly	Daily pattern analysis
`salesdaily.csv`	~2,190	Daily	Model training (resampled to weekly)
`salesweekly.csv`	302	Weekly	Primary forecasting dataset
`salesmonthly.csv`	~72	Monthly	Seasonality analysis

Drug Categories (ATC Classification)

Code	Category	Description
M01AB	Anti-inflammatory	Acetic acid derivatives (e.g., Diclofenac)
M01AE	Anti-inflammatory	Propionic acid derivatives (e.g., Ibuprofen)
N02BA	Analgesics	Salicylic acid derivatives (e.g., Aspirin)

Code	Category	Description
N02BE	Analgesics	Pyrazolones and Anilides (e.g., Paracetamol)
N05B	Psycholeptics	Anxiolytic drugs
N05C	Psycholeptics	Hypnotics and sedatives
R03	Respiratory	Drugs for obstructive airway diseases
R06	Antihistamines	Antihistamines for systemic use

Data Schema

Column	Type	Description
`datum` / `Date`	datetime	Transaction timestamp
`M01AB`	float	Sales volume
`M01AE`	float	Sales volume
`N02BA`	float	Sales volume
`N02BE`	float	Sales volume
`N05B`	float	Sales volume
`N05C`	float	Sales volume
`R03`	float	Sales volume
`R06`	float	Sales volume
`Year`	int	Year
`Month`	int	Month (1-12)
`Hour`	int	Hour (0-23)
`Weekday Name`	string	Day of week

🔬 Methodology

End-to-End Pipeline (Per Drug Category)

```
┌─────────────────────────────────────────────────────────────────────┐
│                         DATA PREPROCESSING                          │
│  • Load salesdaily.csv                                              │
│  • Drop metadata columns (Year, Month, Hour, Weekday Name)          │
│  • Convert datum to datetime index                                  │
│  • Resample daily → weekly aggregation                              │
└─────────────────────────────────────────────────────────────────────┘
                                   │
                                   ▼
```

```
┌──────────────────────────────────────────────────────────────────────┐
│                       EXPLORATORY DATA ANALYSIS                      │
│   📊 Visualization        │ 📈 Decomposition      │ 🔗 Autocorrelation │
│   • Daily/Monthly plots  │ • Trend extraction    │ • ACF plots       │
│   • Pandas Profiling     │ • Seasonality         │ • PACF plots      │
│   • Box plots            │ • Residuals           │ • Lag analysis    │
└──────────────────────────────────────────────────────────────────────┘
                                    │
                                    ▼
┌──────────────────────────────────────────────────────────────────────┐
│                         STATIONARITY TESTING                         │
│   • Rolling mean/std visualization                                   │
│   • Augmented Dickey-Fuller (ADF) test                               │
│   • KPSS test                                                        │
│   • Detrending & Differencing if needed                              │
└──────────────────────────────────────────────────────────────────────┘
                                    │
                                    ▼
┌──────────────────────────────────────────────────────────────────────┐
│                         TRAIN-TEST SPLIT                             │
│   Training: 2014-01-02 to 2018-01-14                                 │
│   Testing:  2018-01-21 to 2019-10-08                                 │
└──────────────────────────────────────────────────────────────────────┘
                                    │
                      ┌─────────────┼─────────────┐
                      ▼             ▼             ▼
┌─────────────────┐   ┌─────────────────┐   ┌─────────────────┐
│     SARIMA      │   │     PROPHET     │   │       SVR       │
│                 │   │                 │   │                 │
│ • Grid search   │   │ • yearly=True   │   │ • MinMaxScaler  │
│ • (p,d,q)       │   │ • interval=0.95 │   │ • GridSearchCV  │
│ • (P,D,Q,m)     │   │ • freq='M'      │   │ • kernel='rbf'  │
│ • AIC minimize  │   │ • periods=25    │   │ • timesteps=5   │
└─────────────────┘   └─────────────────┘   └─────────────────┘
                      │             │             │
                      └─────────────┼─────────────┘
                                    ▼
┌──────────────────────────────────────────────────────────────────────┐
│                       EVALUATION & COMPARISON                        │
│   • RMSE (Root Mean Squared Error)                                   │
│   • One-step ahead forecasts                                         │
│   • Dynamic forecasts                                                │
│   • Visualization: Actual vs Predicted                               │
└──────────────────────────────────────────────────────────────────────┘
```

🤖 Model Details

1. SARIMA (Seasonal ARIMA)

Seasonal AutoRegressive Integrated Moving Average

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

mod = sm.tsa.statespace.SARIMAX(
    y,
    order=order,
    seasonal_order=seasonal_order,
    enforce_stationarity=False,
    enforce_invertibility=False
)
results = mod.fit()
```

Parameter Selection:

- **Grid Search:** Iterates over all combinations of (p,d,q) and (P,D,Q,m)
- **Criterion:** Minimum AIC (Akaike Information Criterion)
- **Seasonal Period:** m=12 (monthly seasonality in weekly data)

Forecasting Modes:

Mode	Description
`dynamic=False`	One-step ahead (uses actual history up to each point)
`dynamic=True`	Uses forecasted values for subsequent predictions

2. Facebook Prophet

Additive Regression Model

```
m = Prophet(
    interval_width=0.95,
    yearly_seasonality=True
)
m.fit(train)
future = m.make_future_dataframe(periods=25, freq='M')
forecast = m.predict(future)
```

Configuration:

Parameter	Value	Purpose
`interval_width`	0.95	95% confidence interval
`yearly_seasonality`	True	Capture annual patterns
`periods`	25	25-month forecast horizon

Parameter	Value	Purpose
`freq`	'M'	Monthly frequency

Outputs:

- `yhat` — Point forecast
- `yhat_lower` / `yhat_upper` — Confidence bounds
- Trend and seasonality components (via Plotly)

3. Support Vector Regression (SVR)

Machine Learning Approach

```
# Hyperparameter Grid Search
param_grid = {
    'kernel': ['rbf'],
    'gamma': [0.1, 0.01, 0.001],
    'C': [0.1, 1, 10],
    'epsilon': [0.05, 0.1]
}

grid = GridSearchCV(SVR(), param_grid, refit=True, verbose=3)
grid.fit(x_train, y_train)

# Best Parameters (example)
model = SVR(kernel='rbf', gamma=0.01, C=1, epsilon=0.1)
```

Data Preparation:

1. **Scaling:** MinMaxScaler (0-1 range)
2. **Timesteps:** 5 (sliding window approach)
3. **2D Tensor:** Convert to supervised learning format `[X_{t-4}...X_{t-1}] → [y_t]`

📈 Exploratory Data Analysis

Stationarity Tests

Test	Purpose	Interpretation
ADF Test	Unit root detection	P < 0.05 → Stationary
KPSS Test	Trend stationarity	P < 0.05 → Non-stationary
Rolling Stats	Visual inspection	Constant mean/variance → Stationary

Transformations for Stationarity

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

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

Seasonality Analysis

Analysis	Method	Finding
Annual	Box plots by month	Clear seasonal patterns for R03, R06, N02BE
Weekly	Box plots by weekday	Moderate weekly effects
Trend	365-day rolling mean	Increasing trend in most categories

ACF/PACF Analysis

```
ACF (Autocorrelation Function)
─────────────────────────────
• Identifies MA order (q)
• Sharp cutoff at lag k → q = k

PACF (Partial Autocorrelation Function)
───────────────────────────────────────
• Identifies AR order (p)
• Sharp cutoff at lag k → p = k
```

📊 Evaluation Metrics

Metric	Formula	Purpose
RMSE	$\sqrt{\frac{1}{n}\sum(y - \hat{y})^2}$	Primary accuracy metric
MAE	$\frac{1}{n}\sum\|y - \hat{y}\|$	Absolute error
AIC	$2k - 2\ln(\hat{L})$	Model selection (SARIMA)

Model Comparison Framework

```
┌─────────────────────────────────────────────────────────────────┐
│                    PERFORMANCE COMPARISON                       │
├─────────────────┬───────────────┬───────────────┬──────────────┤
│    Category     │    SARIMA     │    Prophet    │     SVR      │
```

```
├──────────────────┼────────────────┼────────────────┼──────────────┤
│      M01AB       │  RMSE: X.XX    │  RMSE: X.XX    │  RMSE: X.XX  │
│      M01AE       │  RMSE: X.XX    │  RMSE: X.XX    │  RMSE: X.XX  │
│      N02BA       │  RMSE: X.XX    │  RMSE: X.XX    │  RMSE: X.XX  │
│       ...        │     ...        │     ...        │     ...      │
└──────────────────┴────────────────┴────────────────┴──────────────┘
```

🚀 Quick Start

Installation

```
# Clone repository
git clone <repository-url>
cd Sales-Prediction-for-Pharmaceutical-Distribution-Companies-by-Time-Series-
Analysis-main

# Create virtual environment
python -m venv pharma_env
source pharma_env/bin/activate  # Windows: pharma_env\Scripts\activate

# Install dependencies
pip install pandas numpy matplotlib seaborn scikit-learn
pip install statsmodels prophet plotly
pip install pandas-profiling ipywidgets
```

Dependencies

```
pandas>=1.3.0
numpy>=1.21.0
matplotlib>=3.4.0
seaborn>=0.11.0
scikit-learn>=1.0.0
statsmodels>=0.13.0
prophet>=1.1.0
plotly>=5.0.0
pandas-profiling>=3.0.0
ipywidgets>=7.6.0
```

Run Notebooks

```
# Launch Jupyter
jupyter notebook
```

```
# Execution Order:
# 1. EDA of the dataset.ipynb (understand data first)
# 2. Applying TS models on [CATEGORY].ipynb (per drug category)
```

📖 Notebook Workflow (Per Category)

Section	Description	Key Code
1. Import & Load	Load libraries, read CSV	`pd.read_csv()`
2. Preprocessing	Date indexing, weekly resampling	`.resample('W').sum()`
3. Visualization	Time series plots	`matplotlib`
4. Decomposition	Trend, seasonality, residuals	`seasonal_decompose()`
5. ACF/PACF	Autocorrelation analysis	`plot_acf()`, `plot_pacf()`
6. Stationarity	ADF, KPSS, rolling stats	`adfuller()`, `kpss()`
7. SARIMA	Grid search, fit, forecast	`SARIMAX()`
8. Prophet	Fit, predict, visualize	`Prophet()`
9. SVR	Scale, grid search, predict	`SVR()`, `GridSearchCV()`
10. Evaluation	RMSE comparison	`mean_squared_error()`

📊 Key Visualizations

1. Time Series Decomposition

```
┌─────────────────────────────────────────┐
│              OBSERVED                   │
├─────────────────────────────────────────┤
│              TREND                      │
├─────────────────────────────────────────┤
│              SEASONALITY                │
├─────────────────────────────────────────┤
│              RESIDUALS                  │
└─────────────────────────────────────────┘
```

2. Seasonal Box Plots

- **Year-wise:** Distribution across years
- **Month-wise:** Monthly seasonal patterns

3. Forecast Comparison

```
───────── Actual Data ─────────
━━━━━━━━━ SARIMA Forecast ━━━━━━━━━
░░░░░░░░░ Prophet Forecast ░░░░░░░░░
▓▓▓▓▓▓▓▓▓ SVR Forecast ▓▓▓▓▓▓▓▓▓
```

4. Prophet Components

- Interactive Plotly trend visualization
- Yearly seasonality curves
- Weekly seasonality patterns

💡 Business Applications

Inventory Planning

Model Output	Business Action
Upward trend	Increase stock procurement
Seasonal peak	Pre-order for high-demand periods
Declining forecast	Reduce orders, prevent overstocking

Category-Specific Insights

Category	Insight	Action
R03, R06	Strong annual seasonality	Seasonal inventory adjustments
N02BE	Predictable patterns	Stable procurement planning
N05B, N05C	High residuals/noise	Maintain safety stock buffer
M01AB, M01AE	Increasing trend	Scale supply chain capacity

⚠️ Limitations & Assumptions

Limitation	Implication
Single pharmacy data	Results may not generalize
6-year historical window	Limited long-term trend capture
Weekly aggregation	Daily patterns smoothed out
No external regressors	Weather, promotions not modeled

Limitation	Implication
Fixed SARIMA order	Manual parameter selection after grid search
SVR scaling	Separate scaler for train/test may introduce bias

---

🔮 Future Enhancements

Enhancement	Description
🔄 **Ensemble Methods**	Combine SARIMA + Prophet + SVR predictions
🌡️ **External Variables**	Weather, economic indicators
🤖 **Deep Learning**	LSTM, Transformer architectures
📊 **Cross-Validation**	Time series CV instead of single split
📈 **Automated Pipelines**	Auto-ARIMA, hyperparameter optimization
🌐 **Multi-Pharmacy**	Aggregate data from multiple locations

🔧 Troubleshooting

Issue	Solution
Prophet installation fails	`pip install prophet` or use conda
Plotly not rendering	Install `nbformat>=4.2.0`
SARIMA convergence warning	Try different (p,d,q) orders
Memory error with profiling	Use smaller data sample
Date parsing errors	Ensure `datum` column is datetime

🧰 Tech Stack

Category	Technologies
Language	Python 3.9
Data Processing	pandas, numpy
Visualization	matplotlib, seaborn, plotly
Statistical Models	statsmodels (SARIMA)
ML Framework	scikit-learn (SVR)
Time Series	Facebook Prophet
Profiling	pandas-profiling
Environment	Jupyter Notebook

📚 References

- statsmodels SARIMAX
- Facebook Prophet
- scikit-learn SVR
- ADF Test
- KPSS Test
- Kaggle Dataset by Milan Zdravković

👤 Credits

Project Type: Capstone Project (CIND 820)
Institution: Ryerson University
Data Source: Kaggle — Milan Zdravković

🤝 Contributing

1. Fork the repository
2. Create a feature branch (`git checkout -b feature/enhancement`)
3. Commit changes (`git commit -m 'Add enhancement'`)
4. Push to branch (`git push origin feature/enhancement`)
5. Open a Pull Request

⭐ Star this repo if you find it useful!

Made with 💙 for Pharmaceutical Time Series Analysis