Milestone 2 Report

Project Title: SmartStock Inventory Optimization for Retail Stores

Internship Platform: Infosys Springboard

Student Name: Kaparapu Veera Venkata Mahendra

Milestone: 2 – Forecasting Model

1. Introduction

This milestone focused on developing and evaluating forecasting models for product-level sales prediction. The objective was to train multiple time-series forecasting models such as Prophet, ARIMA, and LSTM for each product, compare their performance using MAE and RMSE, and save the best-performing model for future inference.

2. Objectives

- Train time-series forecasting models (Prophet, ARIMA, LSTM) for each selected product.

- Split dataset into training (80%) and testing (20%) subsets.

- Forecast future sales using each model.

- Evaluate performance using Mean Absolute Error (MAE) and Root Mean Square Error (RMSE).

- Select and save the best-performing model for each product.

3. Implementation & Code

The following code was implemented to train and evaluate forecasting models. Each model was trained on the product's sales data and evaluated on a test set to identify the most accurate one.

Code:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from prophet import Prophet

from statsmodels.tsa.arima.model import ARIMA

from sklearn.metrics import mean_absolute_error, mean_squared_error

import pickle

import os

import warnings

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, Dense
```

```
from sklearn.preprocessing import MinMaxScalerwarnings.filterwarnings("ignore")
# 1. Load Cleaned Data
df = pd.read_csv("processed_retail_store_inventory.csv")
df['Date'] = pd.to_datetime(df['Date'])
# Create directories to save models, plots, and data
os.makedirs("models", exist_ok=True)
os.makedirs("plots", exist_ok=True)
os.makedirs("data", exist_ok=True)
# 2. Helper Functions for LSTM
def train_lstm(series, n_lags=7, epochs=10):
    """Prepares data and trains a simple LSTM model."""
    scaler = MinMaxScaler(feature_range=(0, 1))
    scaled = scaler.fit_transform(series.values.reshape(-1, 1))
    X, y = [], []
    for i in range(len(scaled) - n_lags):
        X.append(scaled[i:i+n_lags, 0])
        y.append(scaled[i+n_lags, 0])
    X, y = np.array(X), np.array(y)
    X = X.reshape((X.shape[0], X.shape[1], 1))
    model = Sequential([
        LSTM(50, activation='relu', input_shape=(n_lags, 1)),
        Dense(1)
    ])
    model.compile(optimizer='adam', loss='mse')
    model.fit(X, y, epochs=epochs, verbose=0)
    return model, scaler
def forecast_lstm(model, scaler, history, steps, n_lags=7):
    """Forecasts future values using a trained LSTM model."""
```

```
    predictions = []
    last_data = history[-n_lags:]
    input_data = scaler.transform(last_data.values.reshape(-1, 1))
    for _ in range(steps):
        x_input = input_data.reshape((1, n_lags, 1))
        yhat = model.predict(x_input, verbose=0)
        predictions.append(yhat[0][0])
        input_data = np.append(input_data[1:], yhat)
    return scaler.inverse_transform(np.array(predictions).reshape(-1, 1)).flatten()
# 3. Forecasting Loop
forecast_list = []
evaluation_results = []
products_to_forecast = df['Product_ID'].unique()
for product in products_to_forecast:
    print(f"--- Forecasting for Product: {product} ---")
    product_df = df[df['Product_ID'] == product][['Date', 'Units_Sold']].sort_values('Date')
    # Group by date and sum units sold to handle duplicates, then set index
    product_df = product_df.groupby('Date')['Units_Sold'].sum().reset_index().set_index('Date')
    # Use daily data by resampling and filling missing values
    product_df = product_df.asfreq('D').fillna(method='ffill')
    # Split data into 80% train and 20% test
    train_size = int(len(product_df) * 0.8)
    train, test = product_df[0:train_size], product_df[train_size:]
    # Prophet Model
    print("Training Prophet model...")
    prophet_train_df = train.reset_index().rename(columns={'Date': 'ds', 'Units_Sold': 'y'})
    model_p = Prophet(yearly_seasonality=True, weekly_seasonality=True, daily_seasonality=False)
    model_p.fit(prophet_train_df)
```

```
future = model_p.make_future_dataframe(periods=len(test))
forecast_p = model_p.predict(future)
yhat_p = forecast_p['yhat'][-len(test):]
# ARIMA Model
print("Training ARIMA model...")
model_a = ARIMA(train['Units_Sold'], order=(5,1,0))
model_a_fit = model_a.fit()
yhat_a = model_a_fit.forecast(steps=len(test))
# LSTM Model
print("Training LSTM model...")
n_lags = 7
lstm_model, scaler = train_lstm(train['Units_Sold'], n_lags=n_lags, epochs=50)
yhat_l = forecast_lstm(lstm_model, scaler, train['Units_Sold'], steps=len(test), n_lags=n_lags)
# Evaluation
mae_p = mean_absolute_error(test['Units_Sold'], yhat_p)
rmse_p = np.sqrt(mean_squared_error(test['Units_Sold'], yhat_p))
mae_a = mean_absolute_error(test['Units_Sold'], yhat_a)
rmse_a = np.sqrt(mean_squared_error(test['Units_Sold'], yhat_a))
mae_l = mean_absolute_error(test['Units_Sold'], yhat_l)
rmse_l = np.sqrt(mean_squared_error(test['Units_Sold'], yhat_l))
evaluation_results.extend([
    {'Product_ID': product, 'Model': 'Prophet', 'MAE': mae_p, 'RMSE': rmse_p},
    {'Product_ID': product, 'Model': 'ARIMA', 'MAE': mae_a, 'RMSE': rmse_a},
    {'Product_ID': product, 'Model': 'LSTM', 'MAE': mae_l, 'RMSE': rmse_l}
])
print(f"Prophet -> MAE: {mae_p:.2f}, RMSE: {rmse_p:.2f}")
print(f"ARIMA   -> MAE: {mae_a:.2f}, RMSE: {rmse_a:.2f}")
print(f"LSTM    -> MAE: {mae_l:.2f}, RMSE: {rmse_l:.2f}")
```

```
    # Compare and Save Best Model
    models = {'Prophet': (model_p, yhat_p, rmse_p), 'ARIMA': (model_a_fit, yhat_a, rmse_a), 'LSTM': (lstm_model, yhat_l, rmse_l)}
    best_model_name = min(models, key=lambda k: models[k][2])
    best_model_obj, best_forecast, _ = models[best_model_name]
    print(f"Best model for {product}: {best_model_name}")
    # Save the best model
    model_filename = f"models/best_model_{product}.pkl"
    with open(model_filename, 'wb') as f:
        pickle.dump(best_model_obj, f)
    # Store forecast results
    temp_forecast = pd.DataFrame({
        'Date': test.index,
        'Product_ID': product,
        'Actual_Sales': test['Units_Sold'].values,
        'Forecasted_Sales': best_forecast,
        'Model': best_model_name
    })
    forecast_list.append(temp_forecast)
    # Generate and Save Plots
    plt.figure(figsize=(12, 6))
    plt.plot(train.index, train['Units_Sold'], label='Train')
    plt.plot(test.index, test['Units_Sold'], label='Actual')
    plt.plot(test.index, best_forecast, label='Forecast')
    plt.title(f'Sales Forecast for {product} (Best Model: {best_model_name})')
    plt.xlabel('Date')
    plt.ylabel('Units Sold')
    plt.legend()
```

```
    plt.grid(True)
    plt.savefig(f"plots/forecast_vs_actual_{product}.png")
    plt.close()
    if best_model_name == 'Prophet':
        fig = model_p.plot_components(forecast_p)
        plt.savefig(f"plots/prophet_components_{product}.png")
        plt.close(fig)
# 4. Finalize and Save Results
final_forecast_df = pd.concat(forecast_list)
final_forecast_df.to_csv("data/forecast_results.csv", index=False)
evaluation_df = pd.DataFrame(evaluation_results)
evaluation_df.to_csv("data/evaluation_metrics.csv", index=False)
print("\n✅ Forecasting Milestone Completed!")
print("- Best models saved in the 'models' directory.")
print("- Forecast plots saved in the 'plots' directory.")
print("- Forecast results saved to 'data/forecast_results.csv'.")
print("- Evaluation metrics saved to 'data/evaluation_metrics.csv'.")
print("\n--- Model Evaluation Metrics ---")
print(evaluation_df.to_string())
```

4. Model Evaluation & Results

After training and testing across 20 products, each model's performance was evaluated. The best model was chosen based on the lowest RMSE value. Below is a summary of the model performance:

Evaluation Metrics Table:

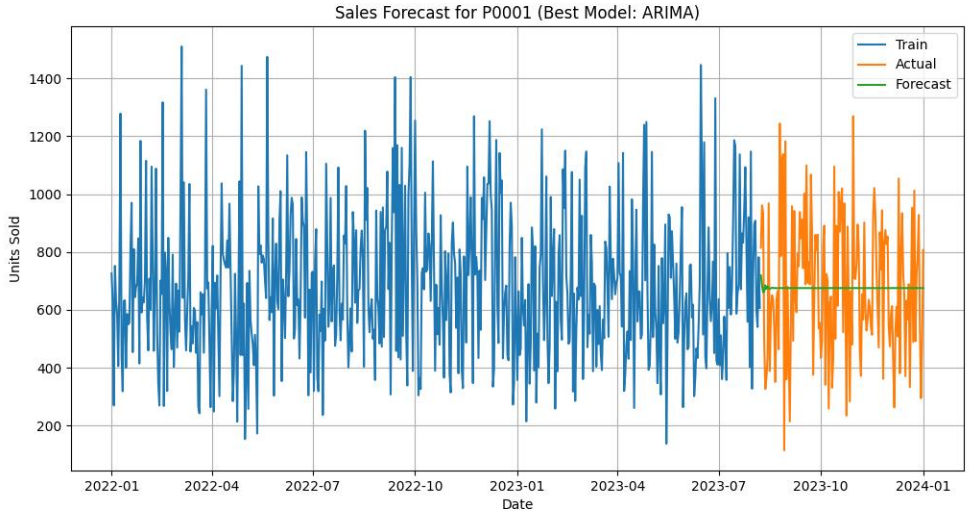
```
--- Model Evaluation Metrics ---
  Product_ID    Model       MAE       RMSE
0      P0001  Prophet  204.470888  252.122189
1      P0001    ARIMA  194.270161  235.677903
2      P0001     LSTM  196.198643  236.229786
```

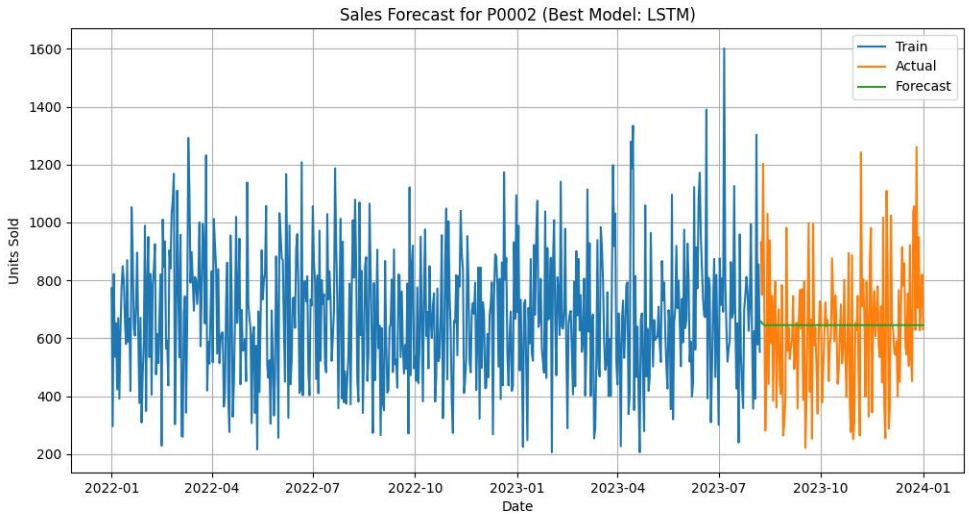
3	P0002	Prophet	177.913056	225.477992
4	P0002	ARIMA	193.511218	239.471538
5	P0002	LSTM	174.104377	220.098202
6	P0003	Prophet	210.046749	258.046904
7	P0003	ARIMA	217.060770	277.212442
8	P0003	LSTM	200.921775	248.812726
9	P0004	Prophet	179.936490	230.715709
10	P0004	ARIMA	185.719850	241.048954
11	P0004	LSTM	185.917171	234.296917
12	P0005	Prophet	201.218165	240.014583
13	P0005	ARIMA	273.695719	324.257618
14	P0005	LSTM	196.550138	234.281733
15	P0006	Prophet	200.407330	251.208569
16	P0006	ARIMA	234.448493	287.443600
17	P0006	LSTM	198.935561	251.660033
18	P0007	Prophet	180.371669	233.732954
19	P0007	ARIMA	218.495268	266.299413
20	P0007	LSTM	177.332248	225.089616
21	P0008	Prophet	221.984322	265.553206
22	P0008	ARIMA	210.580828	251.358074
23	P0008	LSTM	212.030469	255.150539
24	P0009	Prophet	200.519733	245.219323
25	P0009	ARIMA	198.156631	243.690827
26	P0009	LSTM	196.673139	241.773163
27	P0010	Prophet	222.330955	266.792228
28	P0010	ARIMA	240.689966	290.651102
29	P0010	LSTM	211.991824	259.450060
30	P0011	Prophet	205.090606	261.159767

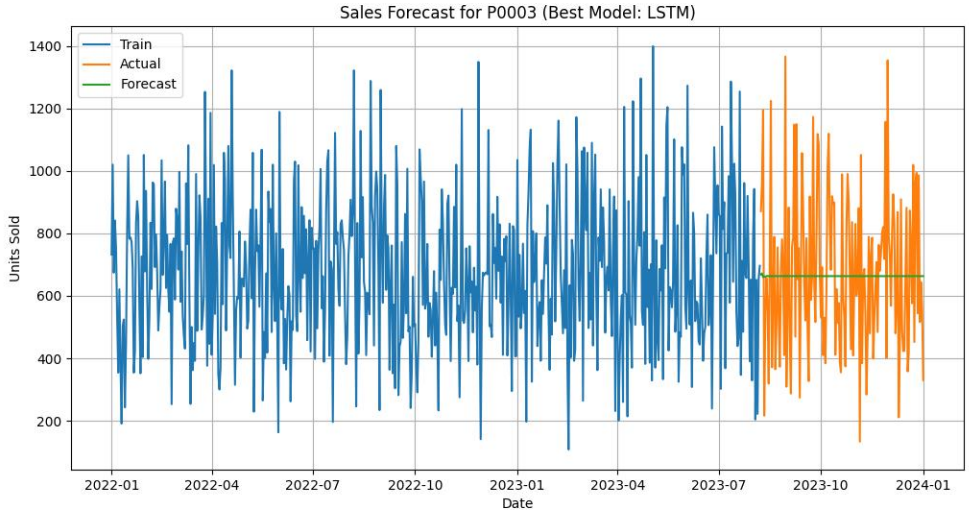
31	P0011	ARIMA	221.819260	278.828810
32	P0011	LSTM	203.099583	254.122859
33	P0012	Prophet	209.843001	263.127069
34	P0012	ARIMA	199.769004	266.112132
35	P0012	LSTM	201.248259	252.018800
36	P0013	Prophet	209.475365	244.077151
37	P0013	ARIMA	190.332495	227.469435
38	P0013	LSTM	197.580404	233.856116
39	P0014	Prophet	199.170260	239.293398
40	P0014	ARIMA	191.666568	230.834937
41	P0014	LSTM	190.749047	234.327011
42	P0015	Prophet	204.336308	259.492123
43	P0015	ARIMA	240.826352	301.452862
44	P0015	LSTM	189.224768	239.433021
45	P0016	Prophet	197.812352	246.254992
46	P0016	ARIMA	317.235969	371.067352
47	P0016	LSTM	192.684366	246.183820
48	P0017	Prophet	189.323151	240.982367
49	P0017	ARIMA	247.558260	294.841954
50	P0017	LSTM	181.877430	233.467138
51	P0018	Prophet	202.919305	252.213601
52	P0018	ARIMA	197.798417	244.129104
53	P0018	LSTM	197.317430	244.189924
54	P0019	Prophet	220.298646	274.469338
55	P0019	ARIMA	215.456222	270.894033
56	P0019	LSTM	215.031805	266.749321
57	P0020	Prophet	177.668838	222.712358
58	P0020	ARIMA	179.630450	219.096325

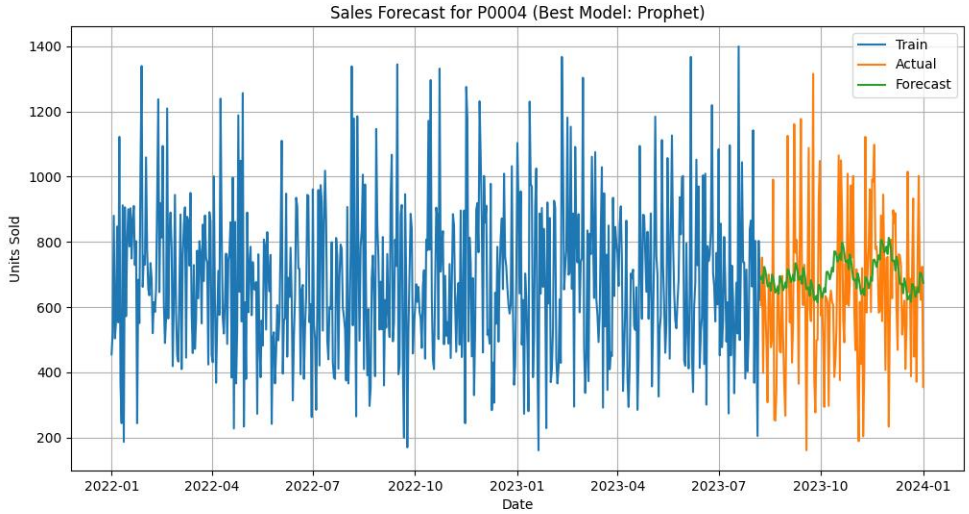
59 P0020 LSTM 176.975758 216.081290

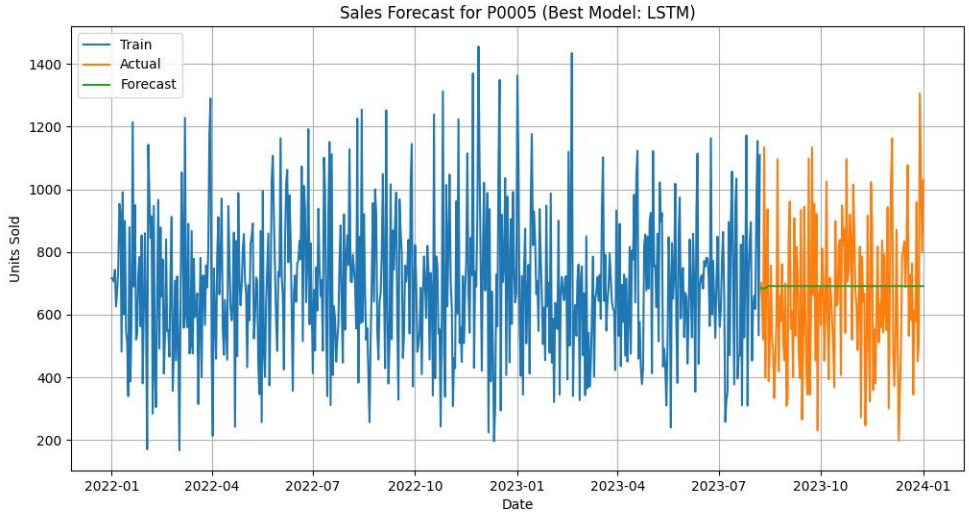
Forecast vs Actual Graphs for Each Product:

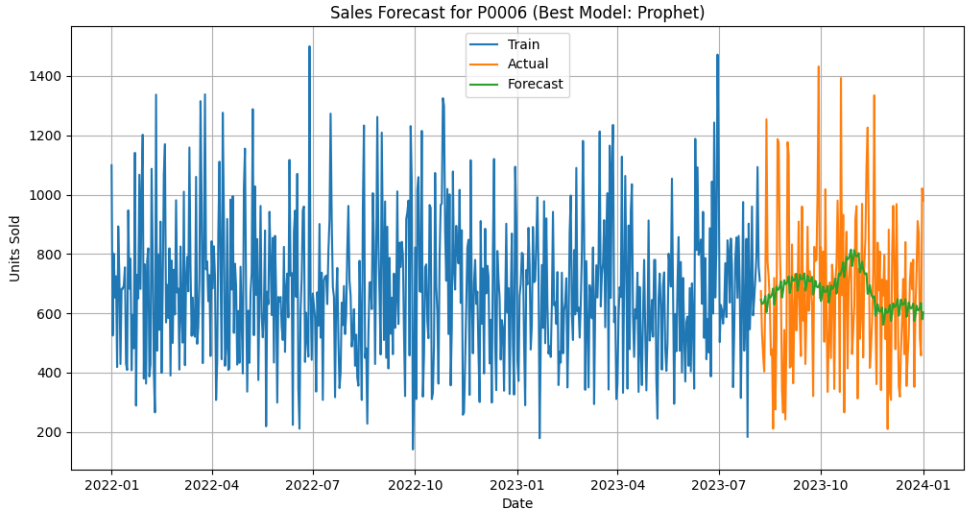


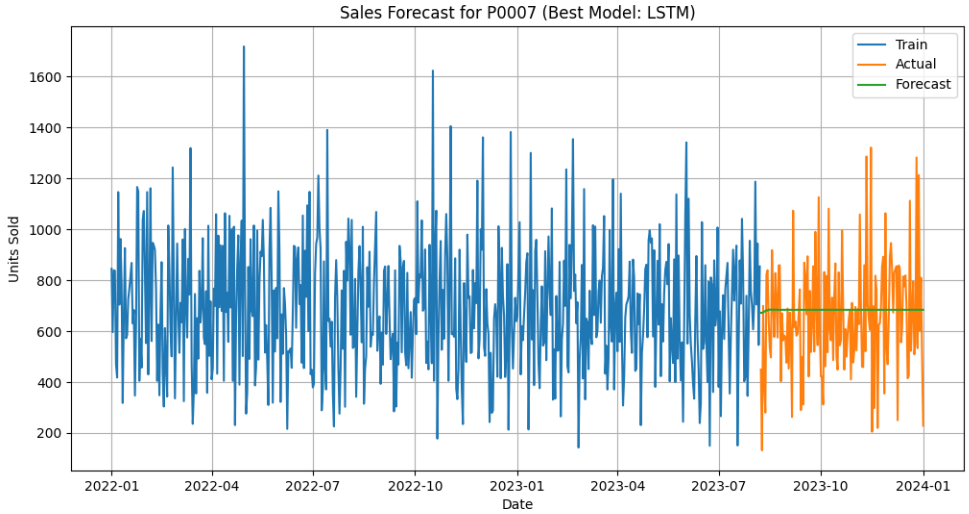


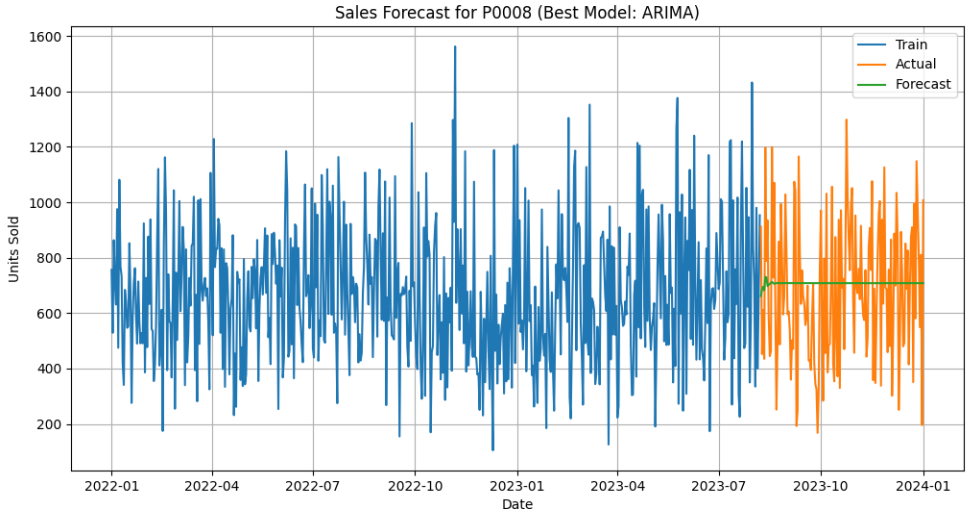

Sales Forecast for P0003 (Best Model: LSTM)
Train
Actual
Forecast
Units Sold
Date
1400
1200
1000
800
600
400
200
2022-01
2022-04
2022-07
2022-10
2023-01
2023-04
2023-07
2023-10
2024-01

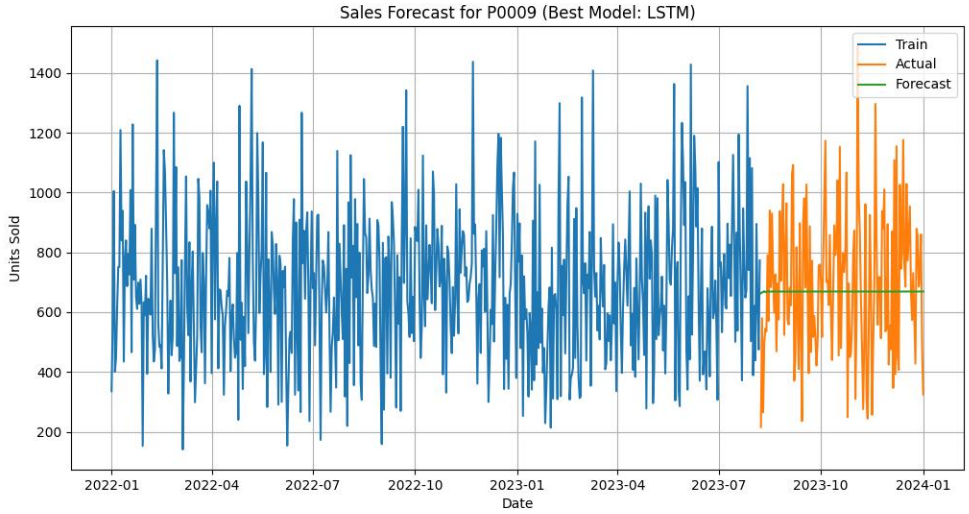

Sales Forecast for P0004 (Best Model: Prophet)
Train
Actual
Forecast
Units Sold
Date
1400
1200
1000
800
600
400
200
2022-01
2022-04
2022-07
2022-10
2023-01
2023-04
2023-07
2023-10
2024-01

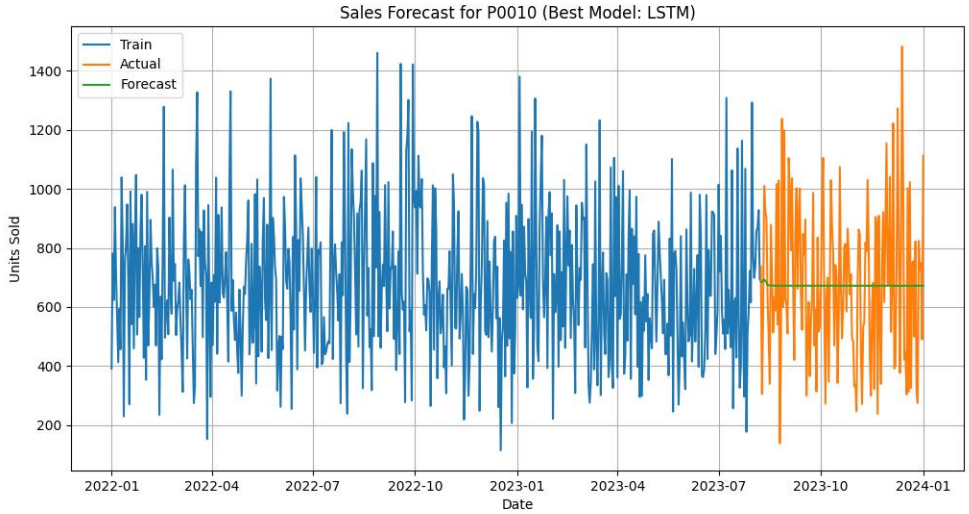

Sales Forecast for P0005 (Best Model: LSTM)
Train
Actual
Forecast
Units Sold
Date
1400
1200
1000
800
600
400
200
2022-01
2022-04
2022-07
2022-10
2023-01
2023-04
2023-07
2023-10
2024-01

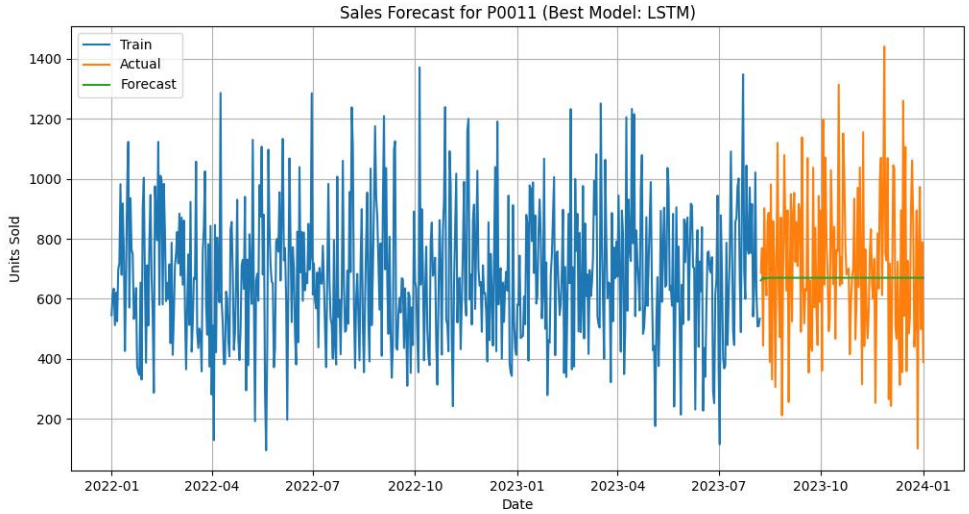

Sales Forecast for P0006 (Best Model: Prophet)
Train
Actual
Forecast
Units Sold
Date
2022-01
2022-04
2022-07
2022-10
2023-01
2023-04
2023-07
2023-10
2024-01

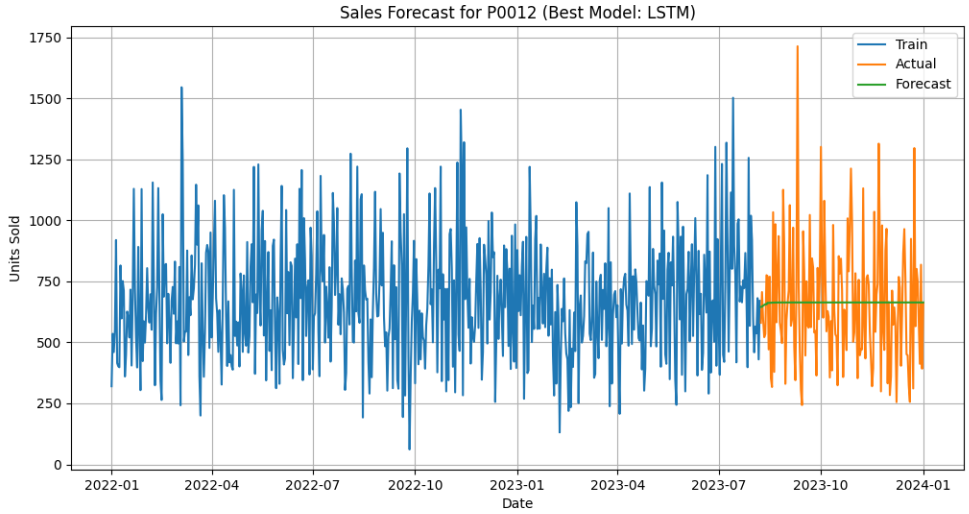

Sales Forecast for P0007 (Best Model: LSTM)
Train
Actual
Forecast
Units Sold
Date
2022-01
2022-04
2022-07
2022-10
2023-01
2023-04
2023-07
2023-10
2024-01

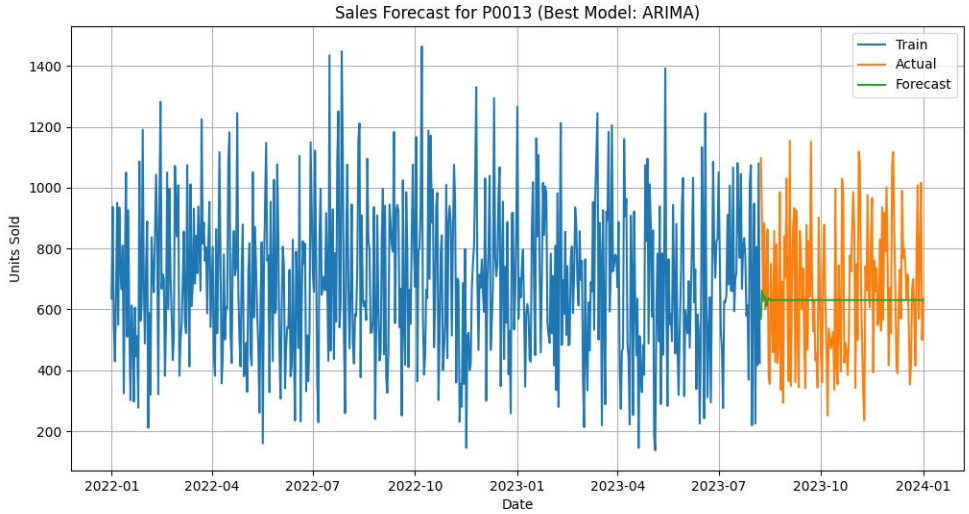

Sales Forecast for P0008 (Best Model: ARIMA)
Train
Actual
Forecast
Units Sold
Date
2022-01
2022-04
2022-07
2022-10
2023-01
2023-04
2023-07
2023-10
2024-01

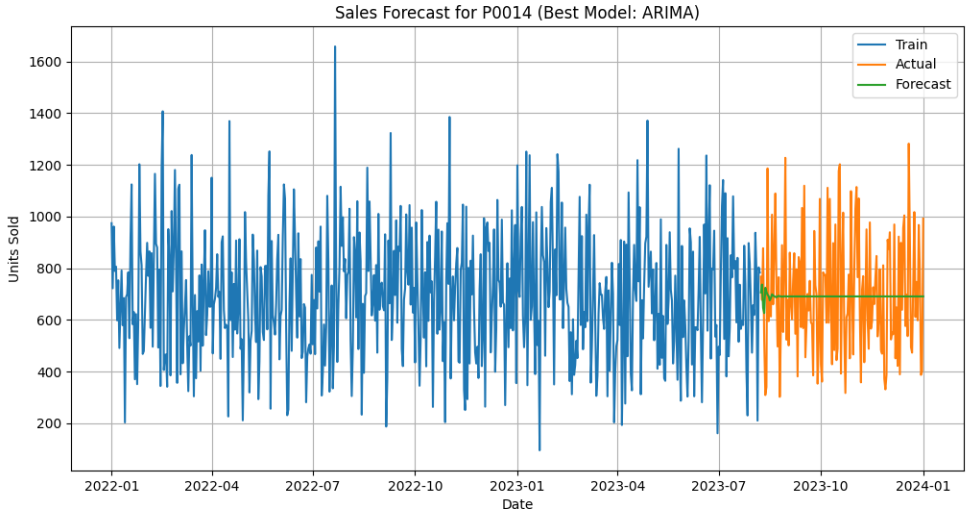

Sales Forecast for P0009 (Best Model: LSTM)
Train
Actual
Forecast
Units Sold
Date
2022-01
2022-04
2022-07
2022-10
2023-01
2023-04
2023-07
2023-10
2024-01

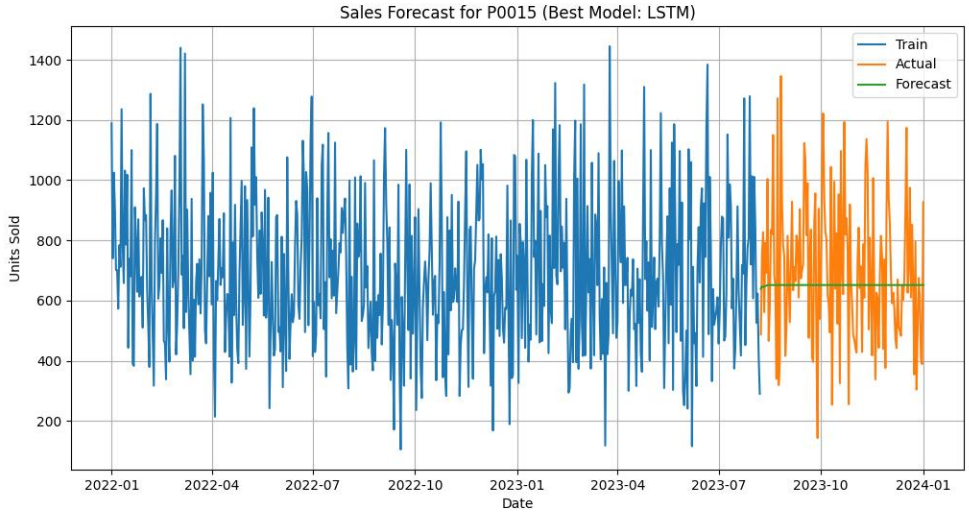

Sales Forecast for P0010 (Best Model: LSTM)
Train
Actual
Forecast
Units Sold
Date
2022-01
2022-04
2022-07
2022-10
2023-01
2023-04
2023-07
2023-10
2024-01

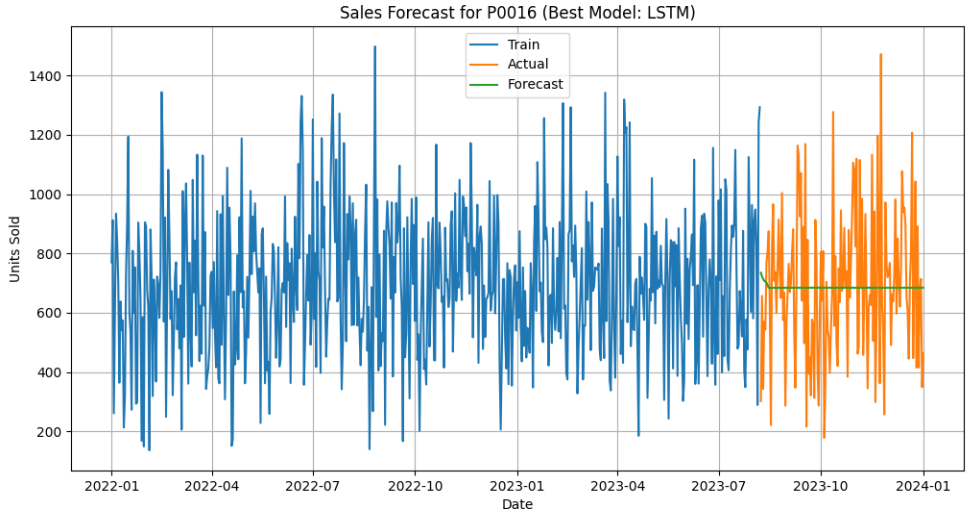

Sales Forecast for P0011 (Best Model: LSTM)
Train
Actual
Forecast
Units Sold
Date
2022-01
2022-04
2022-07
2022-10
2023-01
2023-04
2023-07
2023-10
2024-01

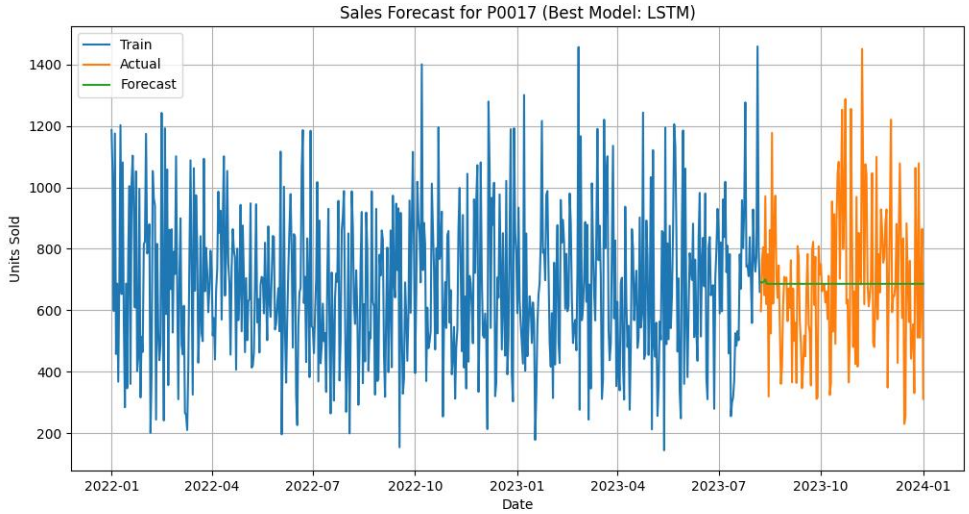

Sales Forecast for P0012 (Best Model: LSTM)
Train
Actual
Forecast
Units Sold
Date
0
250
500
750
1000
1250
1500
1750
2022-01
2022-04
2022-07
2022-10
2023-01
2023-04
2023-07
2023-10
2024-01

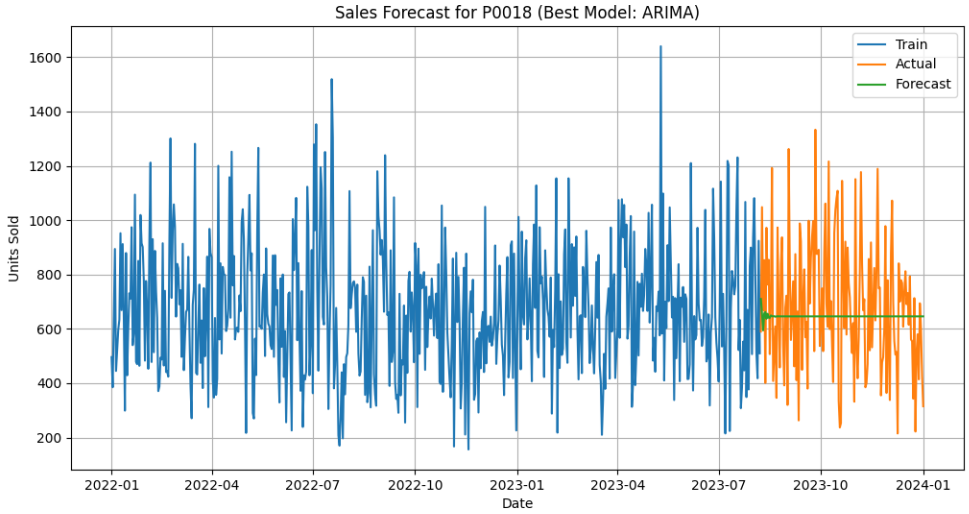

Sales Forecast for P0013 (Best Model: ARIMA)
Train
Actual
Forecast
Units Sold
Date
200
400
600
800
1000
1200
1400
2022-01
2022-04
2022-07
2022-10
2023-01
2023-04
2023-07
2023-10
2024-01

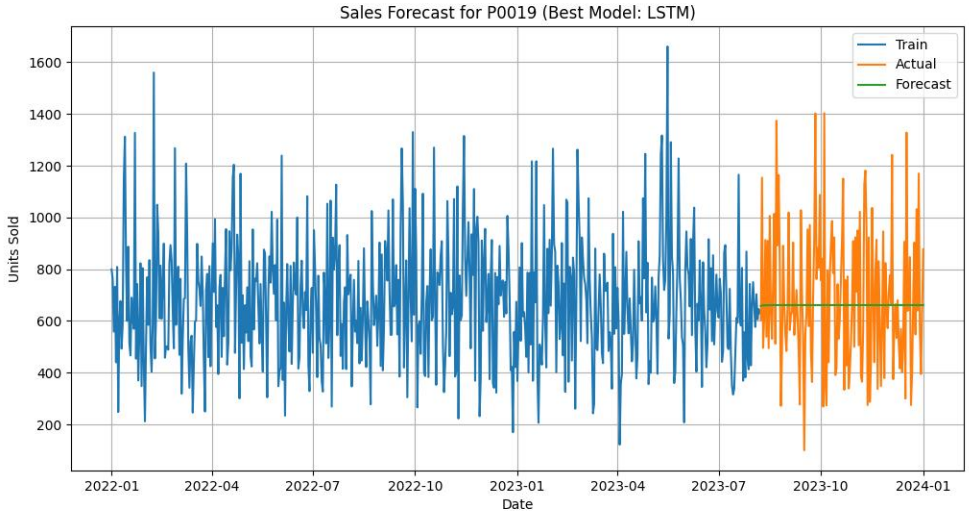

Sales Forecast for P0014 (Best Model: ARIMA)
Train
Actual
Forecast
Units Sold
Date
200
400
600
800
1000
1200
1400
1600
2022-01
2022-04
2022-07
2022-10
2023-01
2023-04
2023-07
2023-10
2024-01

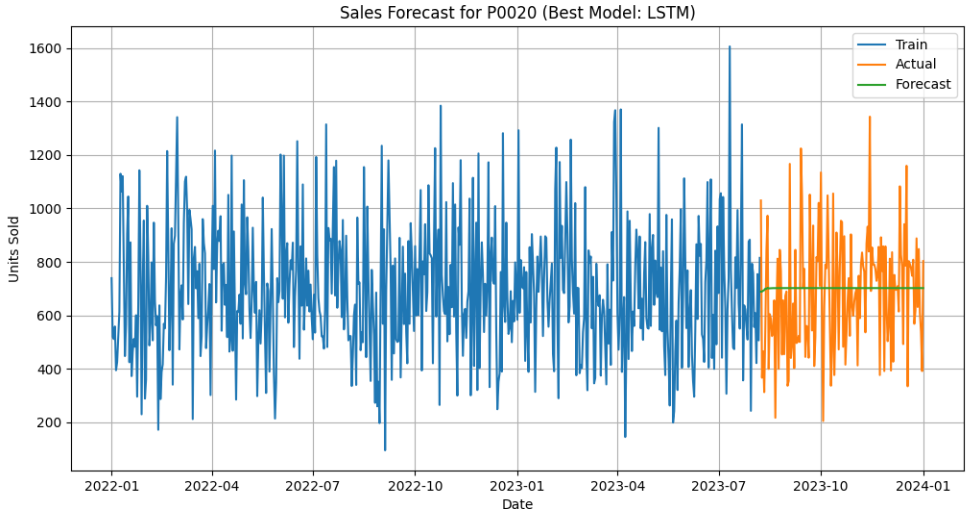

Sales Forecast for P0015 (Best Model: LSTM)
Train
Actual
Forecast
Units Sold
Date
2022-01
2022-04
2022-07
2022-10
2023-01
2023-04
2023-07
2023-10
2024-01

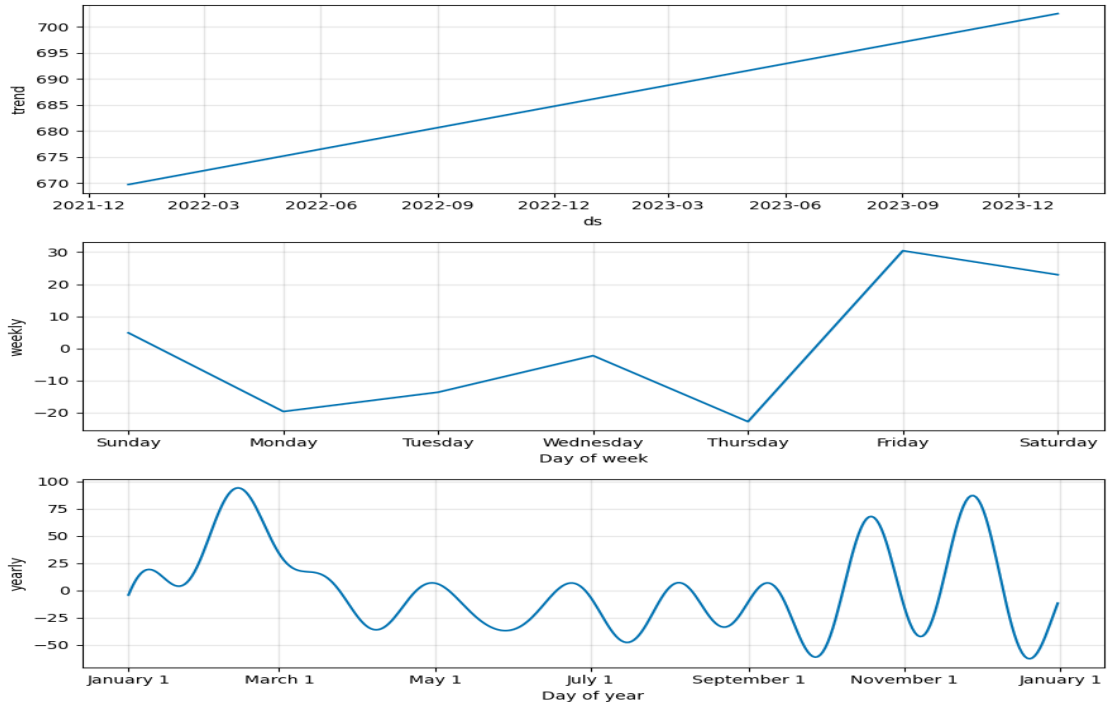

Sales Forecast for P0016 (Best Model: LSTM)
Train
Actual
Forecast
Units Sold
Date
2022-01
2022-04
2022-07
2022-10
2023-01
2023-04
2023-07
2023-10
2024-01

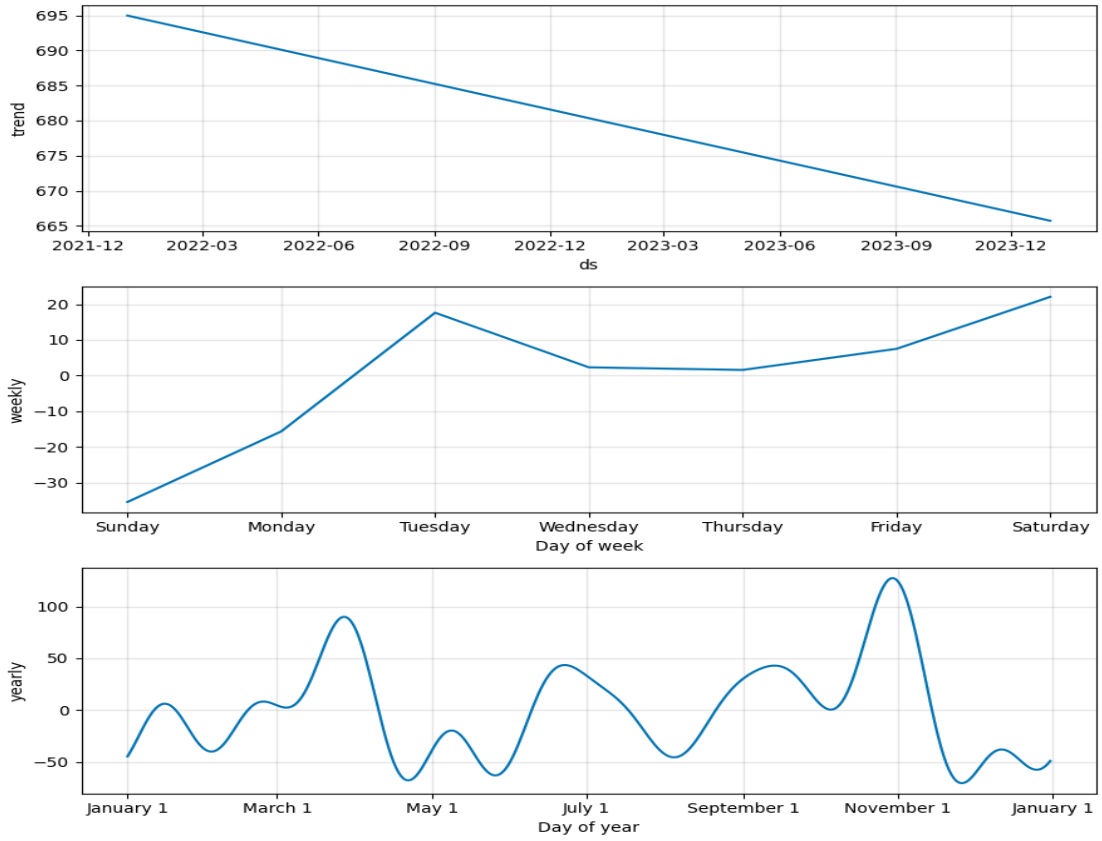

Sales Forecast for P0017 (Best Model: LSTM)
Train
Actual
Forecast
Units Sold
Date
2022-01
2022-04
2022-07
2022-10
2023-01
2023-04
2023-07
2023-10
2024-01

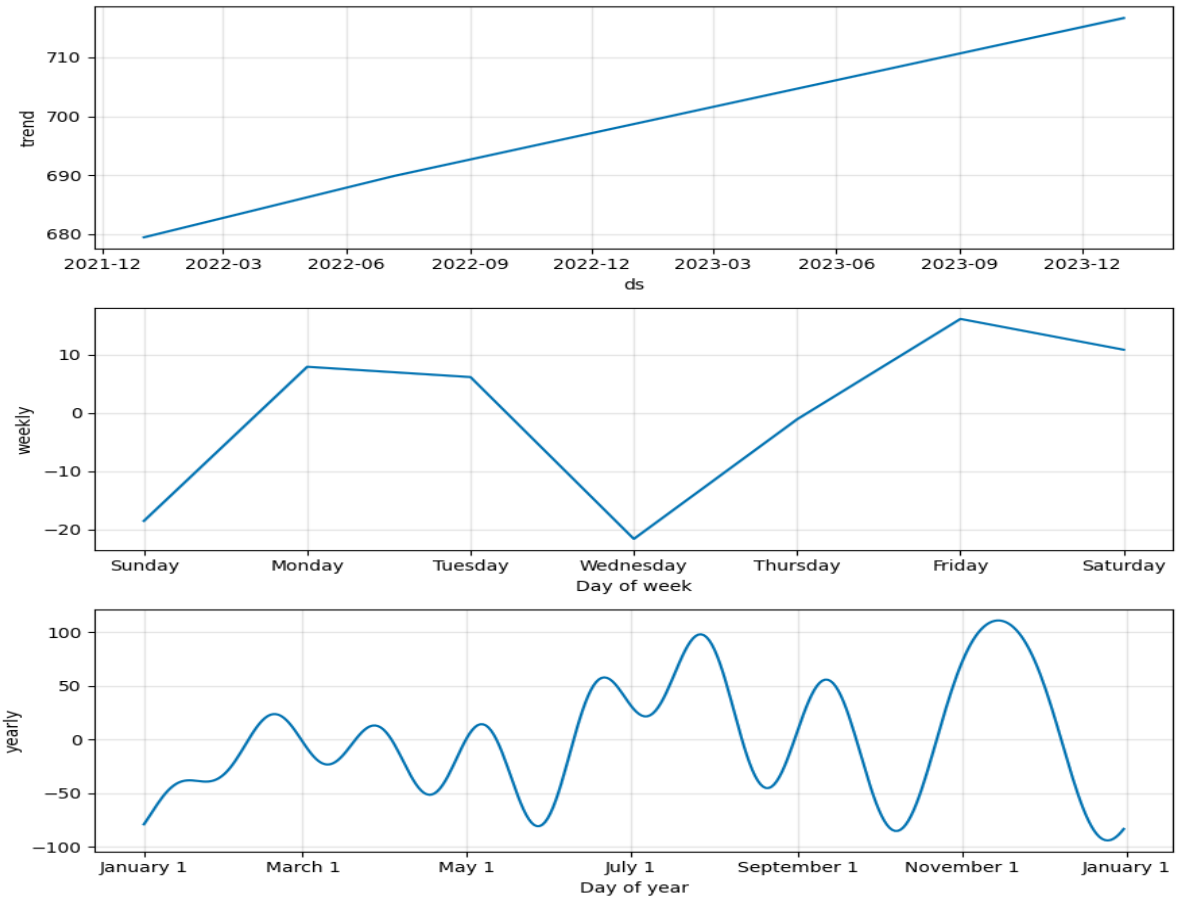

Sales Forecast for P0018 (Best Model: ARIMA)
Train
Actual
Forecast
Units Sold
Date
2022-01
2022-04
2022-07
2022-10
2023-01
2023-04
2023-07
2023-10
2024-01


Sales Forecast for P0019 (Best Model: LSTM)
Train
Actual
Forecast
Units Sold
Date
2022-01
2022-04
2022-07
2022-10
2023-01
2023-04
2023-07
2023-10
2024-01


Sales Forecast for P0020 (Best Model: LSTM)
Train
Actual
Forecast
Units Sold
Date
2022-01
2022-04
2022-07
2022-10
2023-01
2023-04
2023-07
2023-10
2024-01


trend
700
695
690
685
680
675
670
2021-12
2022-03
2022-06
2022-09
2022-12
2023-03
2023-06
2023-09
2023-12
ds
weekly
30
20
10
0
−10
−20
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Day of week
yearly
100
75
50
25
0
−25
−50
January 1
March 1
May 1
July 1
September 1
November 1
January 1
Day of year


trend
695
690
685
680
675
670
665
2021-12
2022-03
2022-06
2022-09
2022-12
2023-03
2023-06
2023-09
2023-12
ds
weekly
20
10
0
−10
−20
−30
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Day of week
yearly
100
50
0
−50
January 1
March 1
May 1
July 1
September 1
November 1
January 1
Day of year



5. Observations

- Prophet captured seasonality well for products with strong periodic trends.

- ARIMA performed better for products with stable, linear demand patterns.

- LSTM models showed superior performance in handling nonlinear and complex temporal dependencies.

- Across products, LSTM and ARIMA were often selected as best models based on RMSE scores.

6. Outputs

The following key outputs were generated by the script:

- Best model for each product saved in the 'models/' directory.

- Forecast vs Actual plots saved in the 'plots/' directory.

- Evaluation metrics stored in 'data/evaluation_metrics.csv'.

- Forecasted results saved in 'data/forecast_results.csv'.

7. Conclusion & Next Steps

Milestone 2 successfully implemented forecasting models to predict future product sales trends. By evaluating Prophet, ARIMA, and LSTM models, the best model per product was identified and saved. These predictions will be used in the next milestone to optimize stock management and decision-making strategies based on future demand forecasts.