Inventory Intelligence: Advanced Inventory Stock Analysis

By: Luis Mireles

Technical Summary of Inventory Demand Forecasting and Optimization Methods

1. I start by loading libraries such as pandas ,numpy ,matplotlib ,pmdarima ,scipy ,matplotlib ,statsmodels.
2. I read the dataset from the CSV file and convert the 'InvoiceDate' column to a datetime format.
3. I filter out rows and compute the frequency of each StockCode.
4. I identify the most common StockCode and filter the dataset to only include transactions with this StockCode.
5. I compute the mean and standard deviation of the quantity and filter out outliers (values greater than 3 standard deviations away from the mean).
6. I aggregate the data on a weekly basis, calculating the sum of quantity for each week.
7. I determine if a given period is seasonal by checking if its quantity is greater than 2 standard deviations away from the mean demand.
8. I check the stationarity of the time series data and apply differencing if necessary.
9. I calculate the autocorrelation function (ACF), partial autocorrelation function (PACF), and Ljung-Box Q (LBQ) values.
10. I find the best (p, d, q) combination for the ARIMA model based on the Akaike Information Criterion (AIC).
11. I fit the ARIMA model with the best order and compare it to an ARIMA model with seasonality and an ARIMA model with a manually chosen order.

12. I forecast the demand for the next two weeks using the ARIMA model with seasonality.
13. I calculate the safety stock and reorder point based on historical demand and recommend an inventory management strategy.
14. I plot the historical and forecasted demand, along with the upper and lower bounds of the forecasted demand.
15. Inventory Stock Analysis Executive Summary is created.

The output of this code provides valuable insights and recommendations for inventory management, which can help businesses optimize their stock levels and reduce costs.

In [83]:
```
import pandas as pd
import numpy as np
import warnings
import matplotlib.pyplot as plt
from statsmodels.tsa.stattools import adfuller, acf, pacf, q_stat
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from pmdarima.arima import auto_arima
from statsmodels.tsa.arima.model import ARIMA
from scipy.stats import chi2
import matplotlib.dates as mdates
```

```
from statsmodels.tsa.statespace.sarimax import SARIMAX

print("\n\n\n" + "INVENTORY STOCK ANALYSIS".center(100))
print("\n\n" + "BY: LUIS MIRELES".center(100))

file_path = '/Users/fernando/Desktop/Code/Sources/Copy of online_retail_II.csv'
data = pd.read_csv(file_path, encoding='ISO-8859-1')
data['InvoiceDate'] = pd.to_datetime(data['InvoiceDate'])
warnings.filterwarnings("ignore")

data = data[data['Quantity'] > 0]

stockcode_counts = data.groupby('StockCode')['StockCode'].count().reset_index(name='count')

most_common_stockcode = stockcode_counts.loc[stockcode_counts['count'].idxmax()]

print(f"\n\nThe most common StockCode is:\n {most_common_stockcode['StockCode']} with a count of {most_co

data = data[data['Quantity'] > 0]
data = data[data['StockCode'] == most_common_stockcode['StockCode']]

mean = data['Quantity'].mean()
std = data['Quantity'].std()

# Filtering out any values greater than 3 standard deviations away from the mean (outliers)
data = data[data['Quantity'] < mean + 3*std]

# Aggregating data on weekly
weekly_data = data.groupby(pd.Grouper(key='InvoiceDate', freq='W'))['Quantity'].sum().reset_index()
weekly_data.columns = ['ds', 'y']

mean_demand = weekly_data['y'].mean()
std_demand = weekly_data['y'].std()

# Seasonality considerations
def is_seasonal_period(row):
    return (row['y'] > (mean_demand + 2 * std_demand)) or (row['y'] < (mean_demand - 2 * std_demand))

weekly_data['seasonal'] = weekly_data.apply(is_seasonal_period, axis=1)
```

```
print("\n\n WEEKLY DATA WITH SEASONAL COLUMN: \n", weekly_data)

# Check for stationarity and apply differencing if necessary
def adf_test(series):
    result = adfuller(series)
    print(f'ADF Statistic: {result[0]}')
    print(f'p-value: {result[1]}')

print(f"Length of weekly_data: {len(weekly_data)}")
# print(weekly_data)
adf_test(weekly_data['y'])

weekly_data['y_diff'] = weekly_data['y'].diff().dropna()
adf_test(weekly_data['y_diff'].dropna())

# Computing the ACF, PACF, and Ljung-Box Q (LBQ) table
lags = 20
acf_values = acf(weekly_data['y_diff'].dropna(), nlags=lags)
pacf_values = pacf(weekly_data['y_diff'].dropna(), nlags=lags, method='ols')
lbq_values, lbq_pvalues = q_stat(acf_values[1:], len(weekly_data['y_diff'].dropna()))

# Autocorrelation table
acf_table = pd.DataFrame({'Lag': range(1, lags + 1), 'ACF': acf_values[1:], 'T': acf_values[1:] * np.sqrt
print("\nAutocorrelation Table:")
print(acf_table)
plot_acf(weekly_data['y_diff'].dropna())
plt.show()

# Partial Autocorrelation table
pacf_table = pd.DataFrame({'Lag': range(1, lags + 1), 'PACF': pacf_values[1:], 'T': pacf_values[1:] * np.
print("\nPartial Autocorrelation Table:")
print(pacf_table)
plot_pacf(weekly_data['y_diff'].dropna())
plt.show()

# Time Series
plt.figure(figsize=(12, 6))
plt.plot(weekly_data['ds'], weekly_data['y'], label='Daily Quantity Sold')
```

```
plt.xlabel('Date')
plt.ylabel('Quantity Sold')
plt.title('Time Series Plot of Daily Quantity Sold')
plt.legend()
plt.show()

# Function to calculate the AIC of an ARIMA model
def arima_aic(p, d, q, data):
    try:
        model = ARIMA(data, order=(p, d, q))
        model_fit = model.fit()
        return model_fit.aic
    except ValueError:
        return float('inf')

# Finding the best (p, d, q) combination for the ARIMA model based on AIC
min_aic = float('inf')
best_order = None

for p in range(5):
    for d in range(2):
        for q in range(5):
            current_aic = arima_aic(p, d, q, weekly_data['y'])
            if current_aic < min_aic:
                min_aic = current_aic
                best_order = (p, d, q)

print(f"\nBest ARIMA Model: ARIMA{best_order} with AIC: {min_aic}")

# Alternatively, use the auto_arima function to find the best (p, d, q) combination
best_auto_arima_model = auto_arima(weekly_data['y'], seasonal=True, stepwise=True, suppress_warnings=True
print("\n\n Auto Arima Function: \n:", best_auto_arima_model.summary())

# ////////////////////////////////////


#dead model seasonality not being calculated properly, did standard deviations method instead for seasona

# acf_values, ci, qstat, pvalues = acf(weekly_data['y_diff'].dropna(), nlags=52, alpha=0.05, qstat=True,
```

```
# acf = pd.Series(acf_values)
# seasonality_check = acf[acf.abs() == acf.abs().max()].index[0]
# print("\nSeasonality Check:", seasonality_check)


# SARIMA model
# model = SARIMAX(weekly_data['y'], order=best_order, seasonal_order=(0, 1, 0, seasonality_check), enforc
# results = model.fit()


# forecast = results.forecast(steps=2)

# forecast_df = pd.DataFrame({'ds': pd.date_range(start=weekly_data['ds'].max(), periods=2, freq='W'), 'y

# forecast_data = pd.concat([weekly_data, forecast_df], axis=0)

# plt.figure(figsize=(12, 6))
# plt.scatter(forecast_df['ds'], forecast_df['y'], label='Forecasted')
# plt.plot(weekly_data['ds'], weekly_data['y'], label='Actual')
# plt.xlabel('Date')
# plt.ylabel('Quantity Sold')
# plt.title('Scatter Plot of Forecasted vs Actual Quantity Sold')
# plt.legend()
# plt.show()


# ////////////////////////////////////

#for individual control the own_order model is present based on visual analysis of ACF & PACF
own_order = (2,2,1)
print("\n\n\n Best Order on SelfCheck: \n",own_order)

print("\n\n\n Best Order on AutoCheck: \n",best_order)

# ARIMA MODELS
best_arima_model = ARIMA(weekly_data['y'], order=best_order)
best_arima_fit = best_arima_model.fit()

arima_model_with_seasonal = ARIMA(weekly_data['y'], order=best_order, exog=weekly_data['seasonal'])
```

```
arima_fit_with_seasonal = arima_model_with_seasonal.fit()

own_arima_model = ARIMA(weekly_data['y'], order = own_order)
own_arima_fit = own_arima_model.fit()

# Residuals for each
residuals = best_arima_fit.resid
own_residuals = own_arima_fit.resid
seasonal_residuals = arima_fit_with_seasonal.resid

# All plots of the ACF of the residuals
plt.figure()
plot_acf(residuals)
plt.title("ACF of Residuals(Simple ARIMA)")
plt.show()

plt.figure()
plot_acf(own_residuals)
plt.title("ACF of Residuals(OWN)")
plt.show()

plt.figure()
plot_acf(seasonal_residuals)
plt.title("ACF of Residuals(Seasonal)")
plt.show()

#Table calues for all 3

residuals_acf = acf(residuals, nlags=lags)[1:]

lbq_test_values, lbq_test_pvalues = q_stat(residuals_acf, len(residuals))
lbq_test_table = pd.DataFrame({'Lag': range(1, lags + 1), 'ACF': residuals_acf, 'LBQ': lbq_test_values, '
print("\nLjung-Box Test and ACF on Residuals(Simple Arima):")
print(lbq_test_table)

residuals_acf_own = acf(own_residuals, nlags=lags)[1:]

lbq_test_values_own, lbq_test_pvalues_own = q_stat(residuals_acf_own, len(own_residuals))
lbq_test_table_own = pd.DataFrame({'Lag': range(1, lags + 1), 'ACF': residuals_acf_own, 'LBQ': lbq_test_v
```

```
print("\nLjung-Box Test and ACF on Residuals(OWN):")
print(lbq_test_table_own)

residuals_acf_s = acf(seasonal_residuals, nlags=lags)[1:]

lbq_test_values_s, lbq_test_pvalues_s = q_stat(residuals_acf_s, len(seasonal_residuals))
lbq_test_table_s = pd.DataFrame({'Lag': range(1, lags + 1), 'ACF': residuals_acf_s, 'LBQ': lbq_test_value
print("\nLjung-Box Test and ACF on Residuals(Seasonal):")
print(lbq_test_table_s)



####################

# arima_fit_with_seasonal
# best_arima_fit
# own_arima_fit

#Forecast
future_steps = 2
exog_insert = weekly_data.loc[weekly_data.index[-future_steps:], 'seasonal']
forecast = arima_fit_with_seasonal.forecast(steps=future_steps, exog= exog_insert)
# forecast = arima_fit_with_seasonal.forecast(steps=2)
date_range = pd.date_range(weekly_data['ds'].iloc[-1] + pd.Timedelta(weeks=1), periods=forecast.shape[0],
# Combine the date range and forecast
forecast_demand = pd.DataFrame({'ds': date_range, 'yhat': forecast})
forecast_demand['yhat'] = forecast_demand['yhat'].clip(lower=0)

# Inventory Optimization Strategy
safety_stock = 1.5 * weekly_data['y'].std()  # Assuming 1.5 times the standard deviation of historical de
order_point = 2 * weekly_data['y'].mean()  # Assuming 2 week lead time
print("\n\n\n" + "INVENTORY STOCK ANALYSIS REPORT".center(100))
print("\n\n" + "BY: LUIS MIRELES".center(100))

print("\n\n\n\nItem: ", {most_common_stockcode['StockCode']})
print("Safety Stock: {:.0f}".format(safety_stock))
print("Reorder Point: {:.0f}".format(order_point))
inventory_level = order_point - forecast_demand['yhat'].sum()
print("Projected Inventory Level after 2 weeks: {:.0f}".format(inventory_level))
```

```
if inventory_level <= safety_stock:
    order_quantity = order_point + safety_stock - inventory_level
    print("Place an order for: {:.0f}".format(order_quantity))
else:
    print("No need to place an order in the next 2 weeks.")

print("\n\n Forecast Demand: \n", forecast_demand)
# Plots
forecast_demand_weekly = forecast_demand
recent_data = weekly_data.tail(10)
plt.figure(figsize=(12, 6))
plt.plot(weekly_data['ds'], weekly_data['y'], label='Historical Demand', marker='o', linestyle='--')
plt.plot(forecast_demand_weekly['ds'], forecast_demand_weekly['yhat'], label='Forecasted Demand', marker=
plt.scatter(recent_data['ds'], recent_data['y'])
for index, row in forecast_demand_weekly.iterrows():
    plt.annotate(
        f"{row['yhat']:.0f}",
        (row['ds'], row['yhat']),
        textcoords="offset points",
        xytext=(0, 5),
        ha='center',
        fontsize=9,
        color='blue'
    )
plt.title('Historical & Forecasted Demand')
plt.xlabel('Date')
plt.ylabel('Demand')
plt.legend()
ax = plt.gca()
ax.xaxis.set_major_formatter(mdates.DateFormatter('%m/%d/%y'))
plt.show()


# lower and upper bounds for forecasted demand based on residuals
r_build = arima_fit_with_seasonal.resid
std_build = 0.5* np.std(r_build)

forecast_demand_upper = forecast_demand.copy()
```

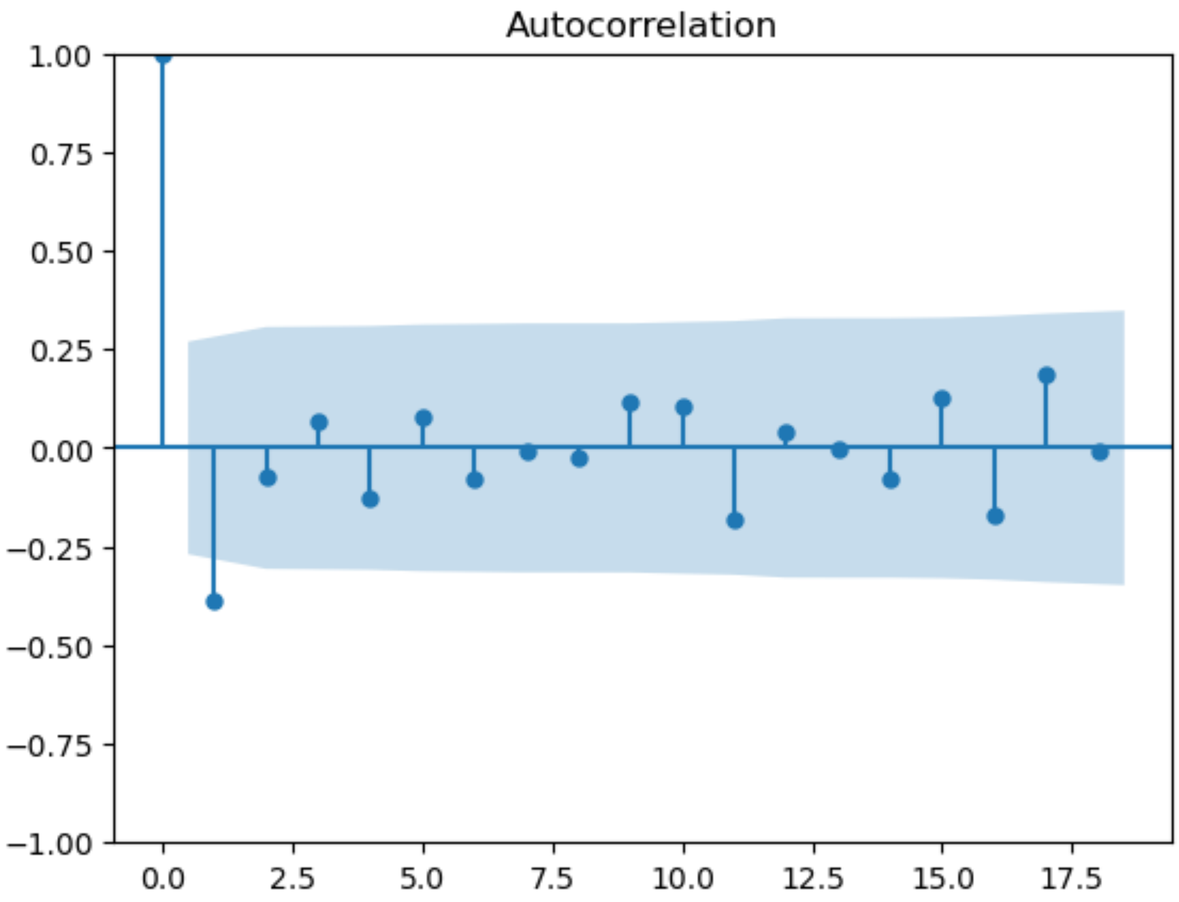
```
forecast_demand_upper['yhat'] += std_build

forecast_demand_lower = forecast_demand.copy()
forecast_demand_lower['yhat'] -= std_build
print("Upper:",forecast_demand_upper['yhat'])
print("Lower:",forecast_demand_lower['yhat'])

forecast_demand_weekly = forecast_demand
recent_data = weekly_data.tail(10)
plt.figure(figsize=(12, 6))
plt.scatter(recent_data['ds'], recent_data['y'], label='Historical Demand')
plt.scatter(forecast_demand_weekly['ds'], forecast_demand_weekly['yhat'], label='Forecasted Demand')
plt.plot(recent_data['ds'], recent_data['y'])
plt.plot(forecast_demand['ds'], forecast_demand['yhat'])
plt.plot(forecast_demand['ds'], forecast_demand_upper['yhat'], label='Upper Bound')
plt.plot(forecast_demand['ds'], forecast_demand_lower['yhat'], label='Lower Bound')
plt.plot([recent_data['ds'].iloc[-1], forecast_demand['ds'].iloc[0]], [recent_data['y'].iloc[-1], forecas
for index, row in forecast_demand_weekly.iterrows():
    plt.annotate(
        f"{row['yhat']:.0f}",
        (row['ds'], row['yhat']),
        textcoords="offset points",
        xytext=(0, 5),
        ha='center',
        fontsize=9,
        color='blue'
    )
plt.title('Recent Historical & Forecasted Demand')
plt.xlabel('Date')
plt.ylabel('Demand')
plt.legend()
ax = plt.gca()
ax.xaxis.set_major_formatter(mdates.DateFormatter('%m/%d/%y'))
plt.show()
```

INVENTORY STOCK ANALYSIS

```
                                BY: LUIS MIRELES


The most common StockCode is:
 85123A with a count of 2270.



 WEEKLY DATA WITH SEASONAL COLUMN:
            ds     y  seasonal
0  2010-12-05   986     False
1  2010-12-12  1045     False
2  2010-12-19  1024     False
3  2010-12-26   198     False
4  2011-01-02     0      True
5  2011-01-09   973     False
6  2011-01-16   389     False
7  2011-01-23   661     False
8  2011-01-30   541     False
9  2011-02-06   586     False
10 2011-02-13   517     False
11 2011-02-20   365     False
12 2011-02-27   401     False
13 2011-03-06   380     False
14 2011-03-13   454     False
15 2011-03-20   465     False
16 2011-03-27   365     False
17 2011-04-03   582     False
18 2011-04-10   374     False
19 2011-04-17   474     False
20 2011-04-24   492     False
21 2011-05-01   375     False
22 2011-05-08  1144      True
23 2011-05-15   873     False
24 2011-05-22   783     False
25 2011-05-29   418     False
26 2011-06-05   628     False
```

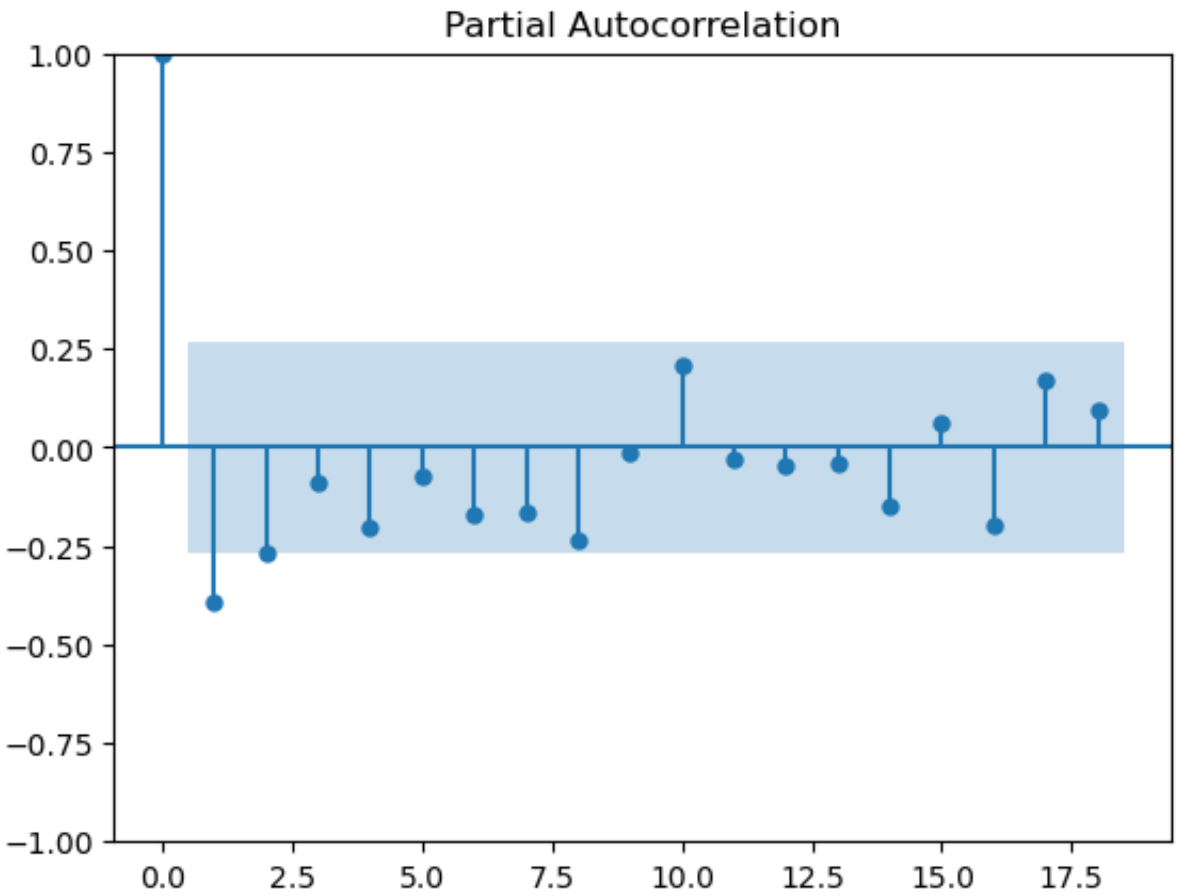
```
27 2011-06-12   389     False
28 2011-06-19   389     False
29 2011-06-26   236     False
30 2011-07-03   259     False
31 2011-07-10   258     False
32 2011-07-17   683     False
33 2011-07-24   719     False
34 2011-07-31   300     False
35 2011-08-07   441     False
36 2011-08-14   466     False
37 2011-08-21   296     False
38 2011-08-28   373     False
39 2011-09-04   394     False
40 2011-09-11   440     False
41 2011-09-18   658     False
42 2011-09-25   705     False
43 2011-10-02   402     False
44 2011-10-09   165     False
45 2011-10-16   679     False
46 2011-10-23   231     False
47 2011-10-30   444     False
48 2011-11-06   712     False
49 2011-11-13   575     False
50 2011-11-20  1322      True
51 2011-11-27   898     False
52 2011-12-04   726     False
53 2011-12-11   649     False
Length of weekly_data: 54
ADF Statistic: -5.501740915673518
p-value: 2.063927536659157e-06
ADF Statistic: -10.612407962191611
p-value: 5.788518496644744e-19

Autocorrelation Table:
    Lag       ACF         T        LBQ  LBQ p-value
0     1 -0.384791 -2.801323   8.300145     0.003964
1     2 -0.071967 -0.523928   8.596175     0.013595
2     3  0.069334  0.504762   8.876438     0.030980
3     4 -0.128026 -0.932041   9.851510     0.043005
```

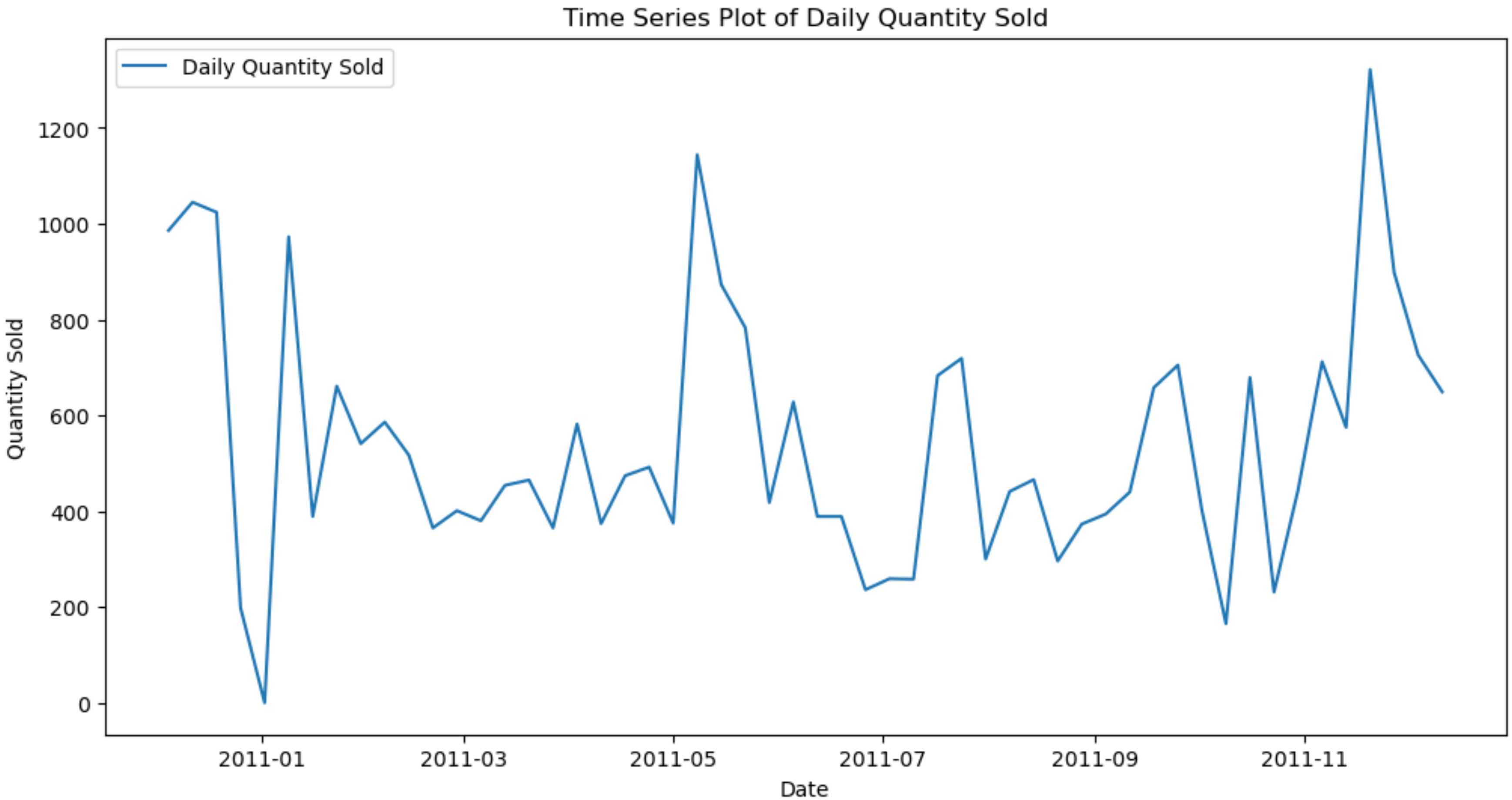
4	5	0.077777	0.566228	10.218881	0.069266
5	6	-0.079130	-0.576074	10.607229	0.101301
6	7	-0.009801	-0.071353	10.613317	0.156398
7	8	-0.022780	-0.165841	10.646931	0.222518
8	9	0.115979	0.844341	11.538072	0.240620
9	10	0.107388	0.781798	12.319849	0.264220
10	11	-0.179982	-1.310290	14.568119	0.203134
11	12	0.038665	0.281486	14.674409	0.259727
12	13	-0.000930	-0.006771	14.674472	0.328113
13	14	-0.079994	-0.582365	15.152759	0.367793
14	15	0.129045	0.939464	16.430195	0.354051
15	16	-0.171981	-1.252043	18.760427	0.281282
16	17	0.185865	1.353115	21.557665	0.202334
17	18	-0.008573	-0.062409	21.563786	0.251929
18	19	-0.011257	-0.081950	21.574650	0.305934
19	20	-0.126474	-0.920742	22.987591	0.289404


Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

Partial Autocorrelation Table:

	Lag	PACF	T
0	1	-0.385152	-2.803951
1	2	-0.262081	-1.907981
2	3	-0.097261	-0.708074
3	4	-0.205982	-1.499571
4	5	-0.060423	-0.439887
5	6	-0.150833	-1.098079
6	7	-0.159027	-1.157737
7	8	-0.217692	-1.584824
8	9	-0.007340	-0.053434
9	10	0.223697	1.628539
10	11	-0.063003	-0.458669
11	12	-0.063888	-0.465110
12	13	-0.013915	-0.101303
13	14	-0.156399	-1.138599
14	15	0.019167	0.139538
15	16	-0.210597	-1.533168
16	17	0.173251	1.261284
17	18	0.126248	0.919099
18	19	-0.007592	-0.055269
19	20	-0.390957	-2.846213

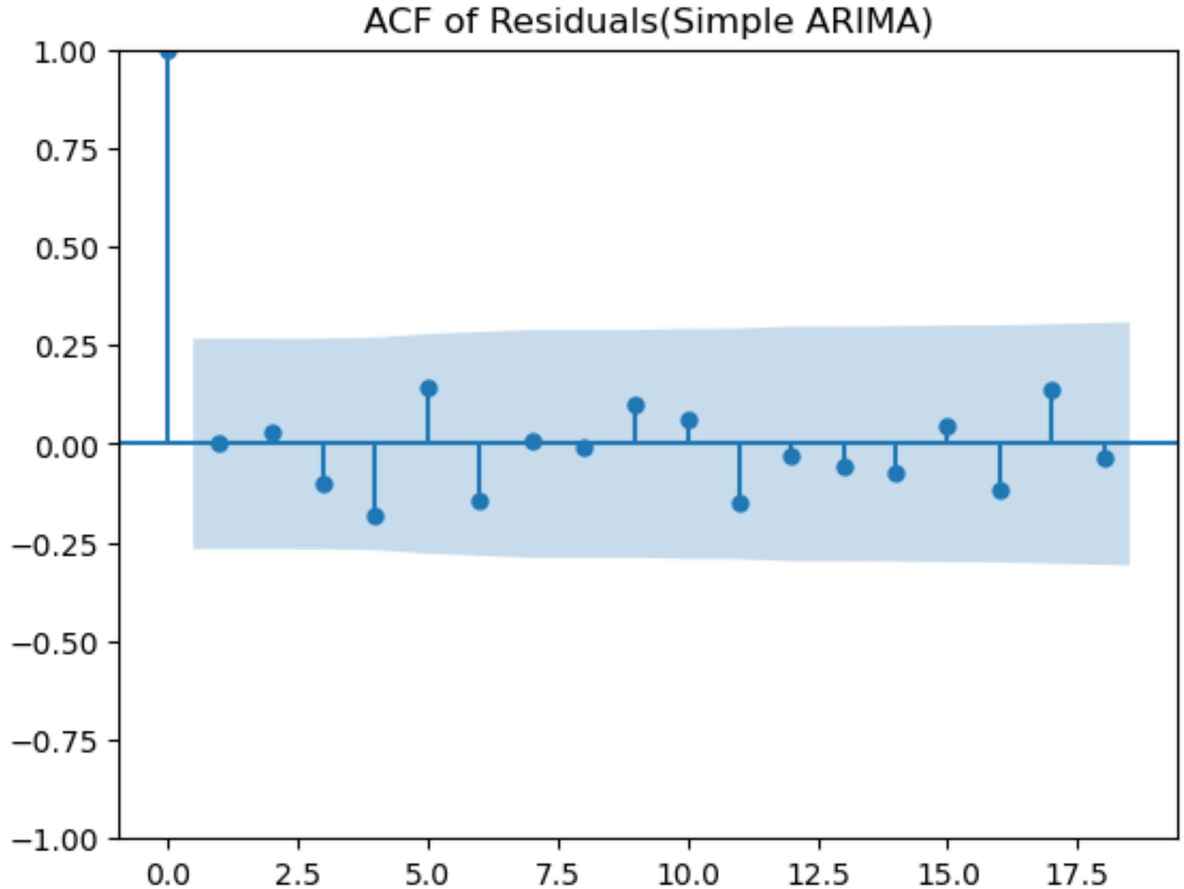

Partial Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5



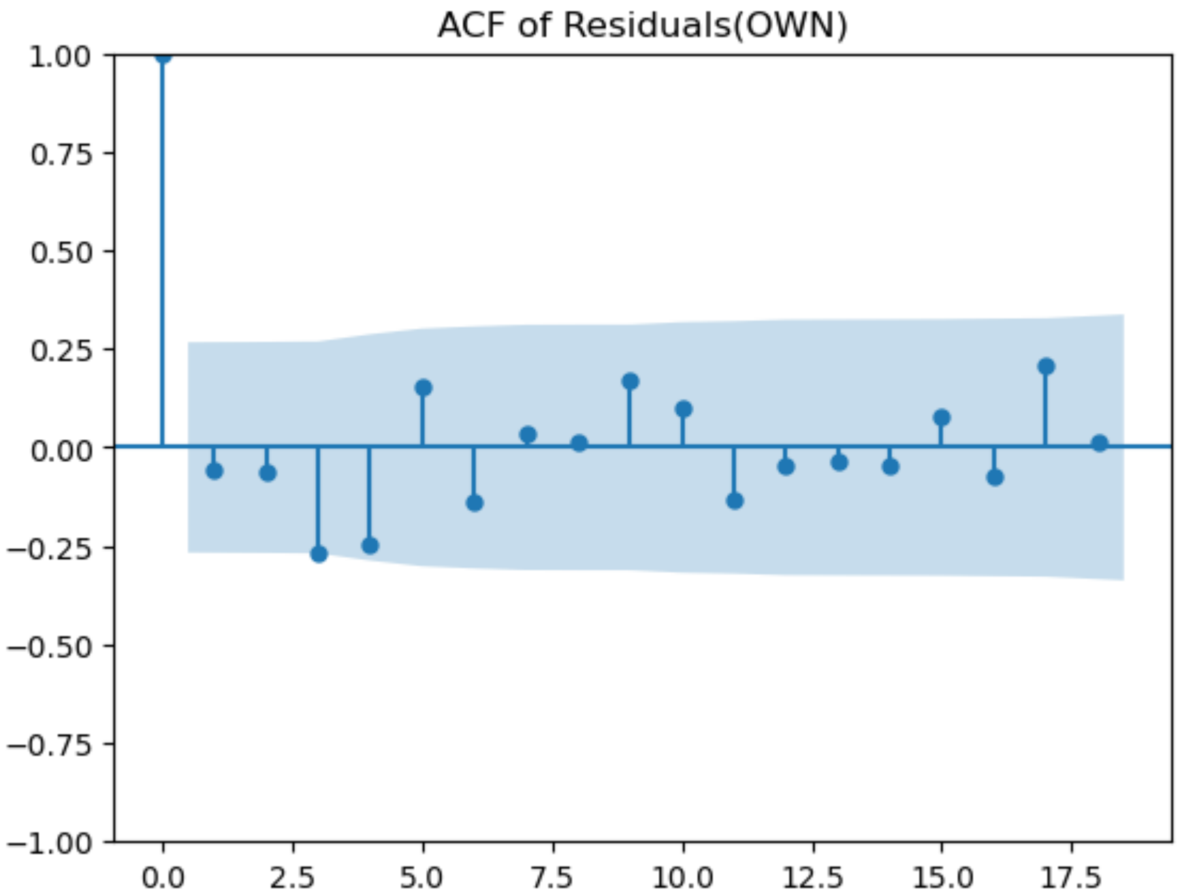

```
Best ARIMA Model: ARIMA(1, 1, 1) with AIC: 746.4759921392756
Performing stepwise search to minimize aic
 ARIMA(2,0,2)(0,0,0)[0] intercept   : AIC=inf, Time=0.08 sec
 ARIMA(0,0,0)(0,0,0)[0] intercept   : AIC=759.197, Time=0.01 sec
 ARIMA(1,0,0)(0,0,0)[0] intercept   : AIC=756.578, Time=0.02 sec
 ARIMA(0,0,1)(0,0,0)[0] intercept   : AIC=757.055, Time=0.03 sec
 ARIMA(0,0,0)(0,0,0)[0]             : AIC=846.684, Time=0.00 sec
 ARIMA(2,0,0)(0,0,0)[0] intercept   : AIC=758.573, Time=0.03 sec
 ARIMA(1,0,1)(0,0,0)[0] intercept   : AIC=758.683, Time=0.03 sec
 ARIMA(2,0,1)(0,0,0)[0] intercept   : AIC=760.551, Time=0.04 sec
 ARIMA(1,0,0)(0,0,0)[0]             : AIC=776.166, Time=0.01 sec
```

```
Best model:  ARIMA(1,0,0)(0,0,0)[0] intercept
Total fit time: 0.260 seconds


 Auto Arima Function:
:                                SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                   54
Model:               SARIMAX(1, 0, 0)   Log Likelihood                -375.289
Date:                Sat, 06 May 2023   AIC                            756.578
Time:                        22:22:39   BIC                            762.545
Sample:                             0   HQIC                           758.880
                                 - 54
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept    387.1800     78.423      4.937      0.000     233.473     540.887
ar.L1          0.2919      0.118      2.474      0.013       0.061       0.523
sigma2      6.358e+04    1.2e+04      5.297      0.000    4.01e+04    8.71e+04
===================================================================================
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):                 8.90
Prob(Q):                              0.91   Prob(JB):                         0.01
Heteroskedasticity (H):               0.78   Skew:                             0.85
Prob(H) (two-sided):                  0.61   Kurtosis:                         4.02
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).


 Best Order on SelfCheck:
 (2, 2, 1)


 Best Order on AutoCheck:
```

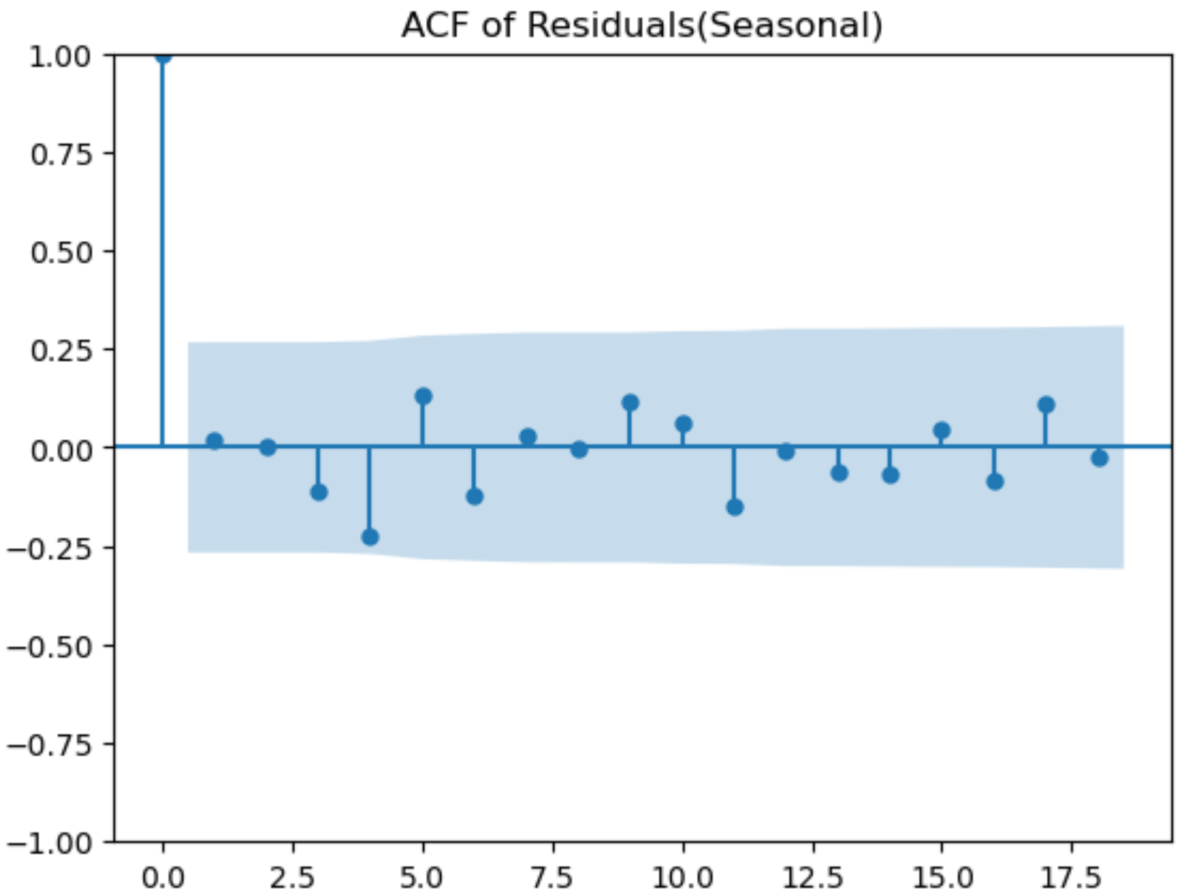
```
 (1, 1, 1)
<Figure size 640x480 with 0 Axes>
```




```
<Figure size 640x480 with 0 Axes>
```




```
<Figure size 640x480 with 0 Axes>
```




```
Ljung-Box Test and ACF on Residuals(Simple Arima):
    Lag       ACF       LBQ  LBQ p-value
0     1  0.002745  0.000430     0.983460
1     2  0.028978  0.049263     0.975669
2     3 -0.098087  0.619737     0.891900
3     4 -0.184038  2.668185     0.614793
4     5  0.144237  3.952100     0.556333
5     6 -0.145587  5.287419     0.507513
6     7  0.009840  5.293649     0.624176
7     8 -0.005590  5.295703     0.725554
8     9  0.097907  5.939874     0.745921
9    10  0.062986  6.212527     0.797103
```

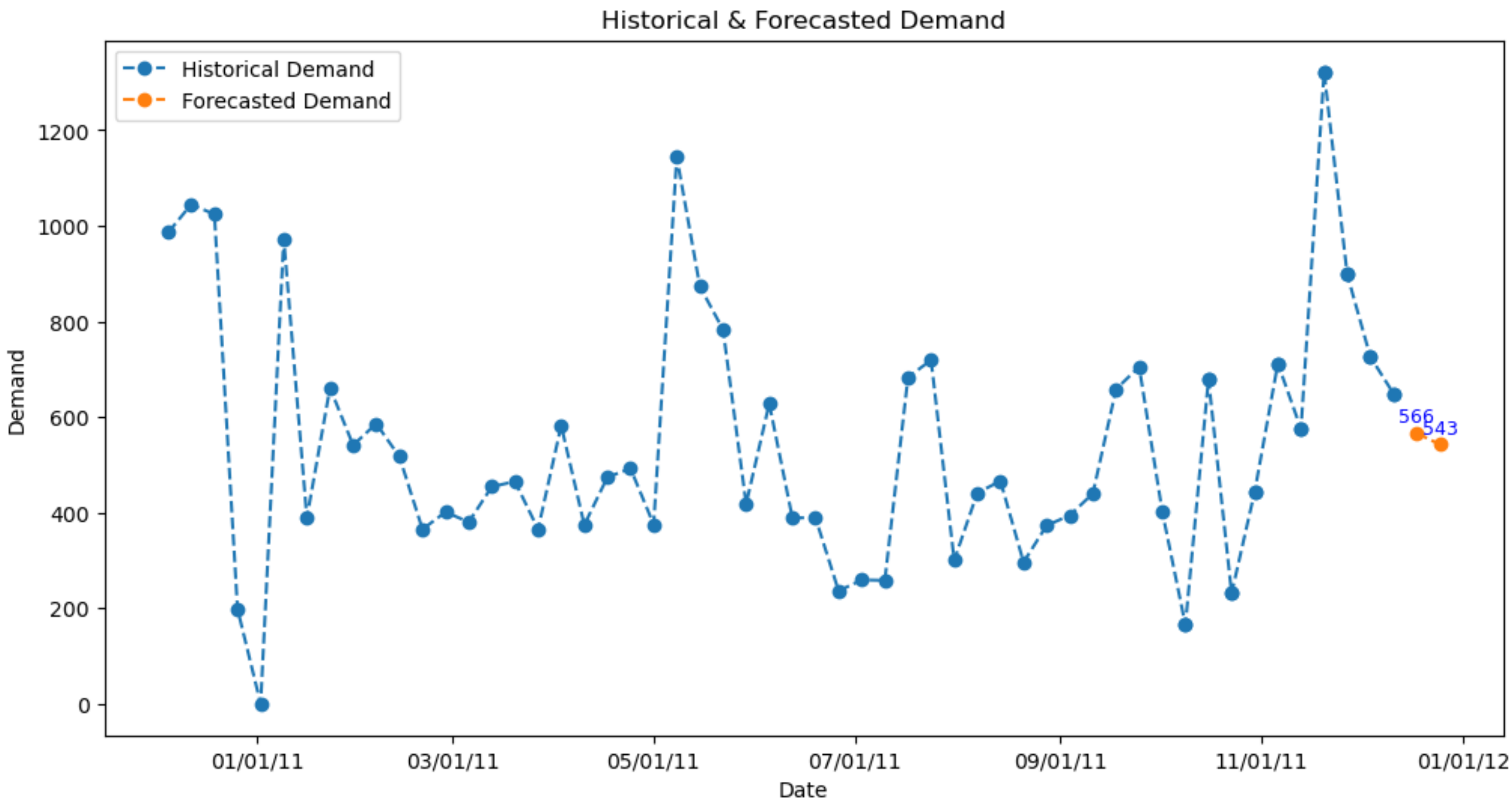
```
10   11 -0.146901   7.730157     0.737253
11   12 -0.029865   7.794373     0.800986
12   13 -0.057973   8.042256     0.840837
13   14 -0.072920   8.444249     0.864923
14   15  0.045422   8.604226     0.897286
15   16 -0.118718   9.725800     0.880550
16   17  0.138745  11.299108     0.840599
17   18 -0.035012  11.402081     0.876522
18   19 -0.049882  11.617066     0.901335
19   20 -0.127028  13.052220     0.875132

Ljung-Box Test and ACF on Residuals(OWN):
    Lag       ACF        LBQ  LBQ p-value
0     1 -0.057951   0.191614     0.661577
1     2 -0.062199   0.416594     0.811966
2     3 -0.268252   4.683341     0.196508
3     4 -0.243690   8.274921     0.082011
4     5  0.153637   9.731646     0.083206
5     6 -0.136283  10.901749     0.091461
6     7  0.034243  10.977192     0.139617
7     8  0.012981  10.988269     0.202365
8     9  0.167759  12.879482     0.168136
9    10  0.101635  13.589414     0.192557
10   11 -0.134695  14.865304     0.188747
11   12 -0.044174  15.005803     0.241119
12   13 -0.036985  15.106693     0.300740
13   14 -0.044803  15.258448     0.360719
14   15  0.078249  15.733211     0.400001
15   16 -0.072263  16.148770     0.442627
16   17  0.210118  19.757113     0.286841
17   18  0.015834  19.778174     0.345446
18   19 -0.045258  19.955150     0.397282
19   20 -0.186099  23.035423     0.287059

Ljung-Box Test and ACF on Residuals(Seasonal):
    Lag       ACF        LBQ  LBQ p-value
0     1  0.017325   0.017126     0.895882
1     2  0.004988   0.018573     0.990757
2     3 -0.111667   0.757942     0.859498
```

```
3      4 -0.226982   3.873908     0.423339
4      5  0.134099   4.983680     0.417875
5      6 -0.119879   5.889048     0.435733
6      7  0.027747   5.938584     0.546939
7      8 -0.004092   5.939685     0.653988
8      9  0.117782   6.871918     0.650453
9     10  0.061966   7.135818     0.712564
10    11 -0.149476   8.707114     0.648907
11    12 -0.008919   8.712841     0.727242
12    13 -0.064179   9.016636     0.771686
13    14 -0.066535   9.351309     0.807909
14    15  0.044528   9.505048     0.849666
15    16 -0.082369  10.044959     0.864270
16    17  0.112181  11.073486     0.852718
17    18 -0.022758  11.116992     0.889326
18    19 -0.030063  11.195079     0.917102
19    20 -0.165529  13.632055     0.848651
```

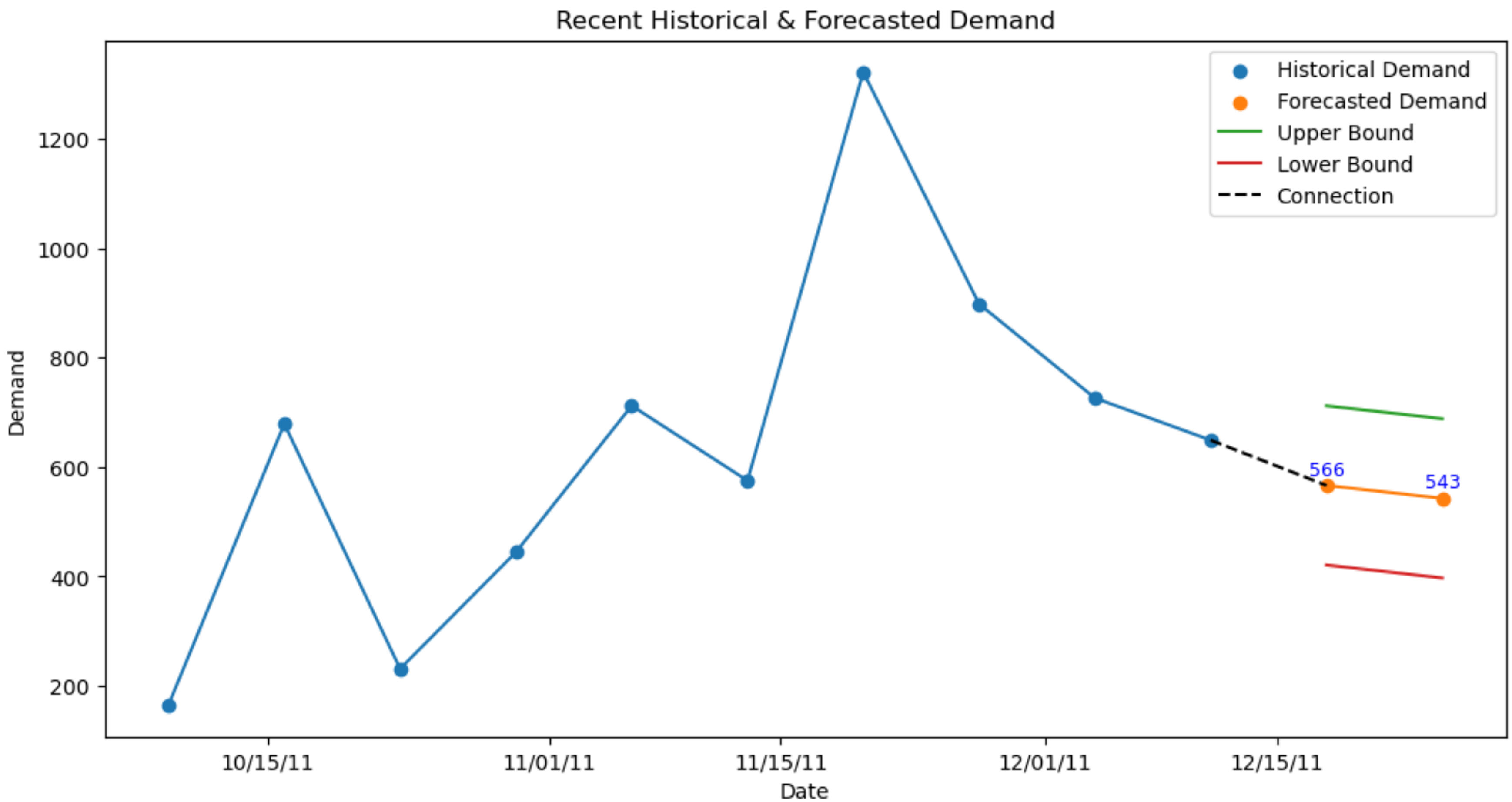
INVENTORY STOCK ANALYSIS REPORT

BY: LUIS MIRELES

```
Item:  {'85123A'}
Safety Stock: 399
Reorder Point: 1085
Projected Inventory Level after 2 weeks: -24
Place an order for: 1508


 Forecast Demand:
            ds        yhat
54 2011-12-18  566.252890
55 2011-12-25  542.626451
```




```
Upper: 54    711.831350
55    688.204911
Name: yhat, dtype: float64
Lower: 54    420.674430
55    397.047991
Name: yhat, dtype: float64
```


Recent Historical & Forecasted Demand
Historical Demand
Forecasted Demand
Upper Bound
Lower Bound
Connection
Demand
1200
1000
800
600
400
200
566
543
10/15/11
11/01/11
11/15/11
12/01/11
12/15/11
Date

Inventory Stock Analysis Executive Summary

By: Luis Mireles

The purpose of this analysis is to forecast the demand for the most frequently sold item **(StockCode: 85123A)** over the next two weeks and provide inventory management recommendations based on the results.

Key Findings:

The historical demand data was analyzed, and the forecast for the next two weeks was generated using an ARIMA model with seasonal adjustments.

The safety stock level was determined as 1.5 times the standard deviation of historical demand, and the reorder point was set at twice the average weekly demand, assuming a 2-week lead time.

Based on the forecast, the **projected inventory level after two weeks is -24**. Given the **safety stock** of **399** and **reorder point** of **1085**, it is recommended to order **1508** in the next two weeks. **Demand for next two weeks is estimated to be 566 and 543.**

The forecasted demand and its upper and lower bounds are visualized in the provided plots, showing the historical and forecasted demand over time. This analysis enables better inventory management and ensures that the stock levels are maintained within the desired range to minimize stockouts and overstocking.

Please note that the demand forecast and inventory recommendations are subject to change with updated data and adjustments to the model parameters. Regular monitoring and analysis are advised to maintain optimal inventory levels.