apps.py > forecast_demand

```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_squared_error
import numpy as np
import streamlit as st
import datetime
from datetime import date,timedelta
import yfinance as yf

# Preprocessing Function
def preprocess_data():
    # Load the data
    df1 = pd.read_csv("data/Transactional_data_retail_01.csv")
    df2 = pd.read_csv("data/Transactional_data_retail_02.csv")
    df_product_info = pd.read_csv("data/ProductInfo.csv")
    df_customer_demographics = pd.read_csv("data/CustomerDemographics.csv")

    # Merge data if necessary
    df = pd.concat([df1, df2])

    # Convert columns to correct datatypes
    df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'],format='mixed')
    df['Quantity'] = pd.to_numeric(df['Quantity'], errors='coerce')

    # Drop missing values if necessary
    df.dropna(subset=['Quantity'], inplace=True)

    # Aggregate data by stock code and transaction date
    df_grouped = df.groupby(['StockCode', 'InvoiceDate']).agg({'Quantity': 'sum'}).reset_index()
```

```python

    # Optional: Add revenue calculation if product price is available
    if 'price' in df_product_info.columns:
        df_grouped = df_grouped.merge(df_product_info[['StockCode', 'price']], on='StockCode', how='left')
        df_grouped['revenue'] = df_grouped['Quantity'] * df_grouped['price']

    return df_grouped

# EDA Function
def perform_eda(df_grouped):
    # Show summary statistics
    st.write("This is the Exploratory Data Analysis (EDA) section.")
    #st.write("Summary Statistics:")
    st.write(df_grouped.describe())

    # Top 10 best-selling products
    top_10_products = df_grouped.groupby('StockCode')['Quantity'].sum().nlargest(10).index

    # Plot sales over time for the top 10 products
    for product in top_10_products:
        product_sales = df_grouped[df_grouped['StockCode'] == product]
        st.write(f"Sales over time for product {product}")
        fig, ax = plt.subplots(figsize=(10, 5))
        sns.lineplot(x='InvoiceDate', y='Quantity', data=product_sales, ax=ax)
        st.pyplot(fig)

# Time Series Forecasting Function
def forecast_demand(df_grouped, stock_codes, num_weeks):
    forecast_data = {}
```

```python
def forecast_demand(df_grouped, stock_codes, num_weeks):
    forecast_data = {}

    for stock_code in stock_codes:
        product_sales = df_grouped[df_grouped['StockCode'] == stock_code].set_index('InvoiceDate')

        # Fit ARIMA model
        model = ARIMA(product_sales['Quantity'], order=(5, 1, 0))
        model_fit = model.fit()

        # Forecast for the next 'num_weeks' weeks
        forecast = model_fit.forecast(steps=num_weeks)

        # Store forecasted values for later use
        forecast_data[stock_code] = forecast

        # Plot historical sales and forecast
        fig, ax = plt.subplots(figsize=(10, 5))
        ax.plot(product_sales.index, product_sales['Quantity'], label='Historical', color='blue')
        ax.plot(pd.date_range(start=product_sales.index[-1], periods=num_weeks, freq='W'), forecast, label='Forecast', color='orange',marker='.')
        ax.set_title(f'Forecast for {stock_code} for next {num_weeks} weeks')
        ax.set_xlabel('Date')
        ax.set_ylabel('Quantity Sold')
        ax.legend()
        st.pyplot(fig)

    return forecast_data

# Error and Evaluation Function
def evaluate_model(df_grouped, stock_codes, num_weeks):
    for stock_code in stock_codes:
        # Filter sales data for the specific stock code
        product_sales = df_grouped[df_grouped['StockCode'] == stock_code].sort_values(by='InvoiceDate').set_index('InvoiceDate')

        # Check for missing values and drop them
        product_sales = product_sales.dropna(subset=['Quantity'])

        # Split data for training (80%) and testing (20%)
        train_size = int(len(product_sales) * 0.8)
        train, test = product_sales['Quantity'][:train_size], product_sales['Quantity'][train_size:]

        # Fit ARIMA model on training data
        try:
            model = ARIMA(train, order=(5, 1, 0))  # Adjust ARIMA parameters as needed or use auto_arima
            model_fit = model.fit()
```

```
            model = ARIMA(train, order=(5, 1, 0))  # Adjust ARIMA parameters as needed or use auto_arima
            model_fit = model.fit()

            # Forecast on the test data length
            predictions = model_fit.predict(start=len(train), end=len(train) + len(test) - 1, dynamic=False)

            # Calculate RMSE (Root Mean Squared Error)
            rmse = np.sqrt(mean_squared_error(test, predictions))

            # Output the result
            print(f"RMSE for {stock_code}: {rmse}")
        except Exception as e:
            print(f"Failed to fit ARIMA model for {stock_code}. Error: {e}")

# Streamlit App with Sidebar and Multi-Selection
def run_app():
    st.title('Demand Forecasting System')

    # Preprocess data
    df_grouped = preprocess_data()

    # Sidebar for input
    st.sidebar.header("User Inputs")

    # Initialize session state for EDA toggle
    if 'show_eda' not in st.session_state:
        st.session_state.show_eda = False

    # EDA Toggle Button
    if st.sidebar.button("Run EDA"):
        st.session_state.show_eda = not st.session_state.show_eda  # Toggle the EDA state

    # Display EDA or main app content based on the toggle state
    if st.session_state.show_eda:
        st.header("Exploratory Data Analysis (EDA)")
        perform_eda(df_grouped)

        # Stock code multi-selection in sidebar
    stock_codes = st.sidebar.multiselect('Select Product Stock Codes', df_grouped['StockCode'].unique(), default=df_grouped['StockCode'].unique()[:1])

        # Slider for number of weeks to forecast
    num_weeks = st.sidebar.slider('Number of weeks to forecast', 1, 15, 5)
```

```
        # Slider for number of weeks to forecast
    num_weeks = st.sidebar.slider('Number of weeks to forecast', 1, 15, 5)

        # Forecasting demand
    st.header("Forecasting Results")
    if len(stock_codes) > 0:
        forecast_data = forecast_demand(df_grouped, stock_codes, num_weeks)

            # Evaluate model performance for selected stock codes
        st.header("Model Evaluation")
        evaluate_model(df_grouped, stock_codes, num_weeks)

            # Download Forecast Data
        st.header("Download Forecast")
        for stock_code in stock_codes:
            forecast_df = pd.DataFrame({'Week': pd.date_range(start=df_grouped[df_grouped['StockCode'] == stock_code]['InvoiceDate'].max(), periods=num_weeks, freq='W'),
                                        'Forecast': forecast_data[stock_code],'StockCode': stock_code})
            csv = forecast_df.to_csv(index=False)
            st.download_button(label=f"Download forecast for {stock_code}", data=csv, file_name=f'forecast_{stock_code}.csv', mime='text/csv')

if __name__ == "__main__":
    run_app()
```

User Inputs

Run EDA

Select Product Stock Codes

84077 × 85123A × 85099B × 21212 × 84879 ×

Number of weeks to forecast

15

1 15

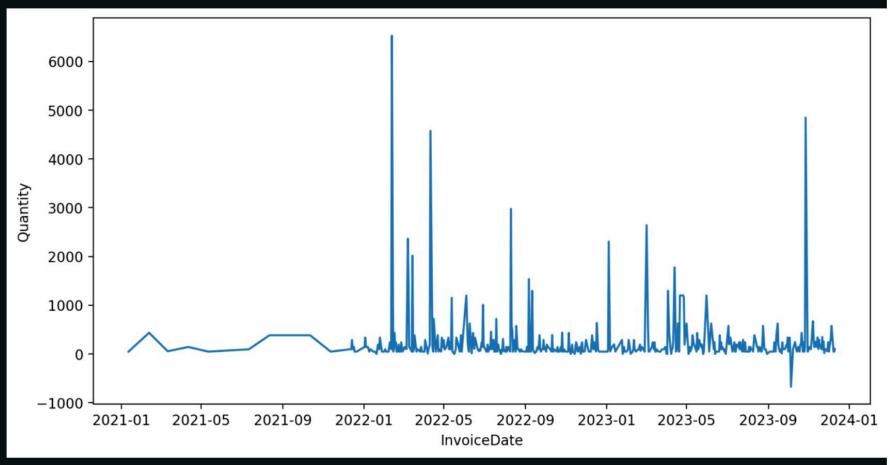
Demand Forecasting System

Exploratory Data Analysis (EDA)

This is the Exploratory Data Analysis (EDA) section.

	InvoiceDate	Quantity
count	542586	542,586
mean	2022-12-14 16:15:11.156572672	19.5517
min	2021-01-12 00:00:00	-19,172
25%	2022-06-17 00:00:00	2
50%	2022-12-10 00:00:00	6
75%	2023-07-03 00:00:00	18
max	2023-12-10 00:00:00	19,152
std	None	124.4585

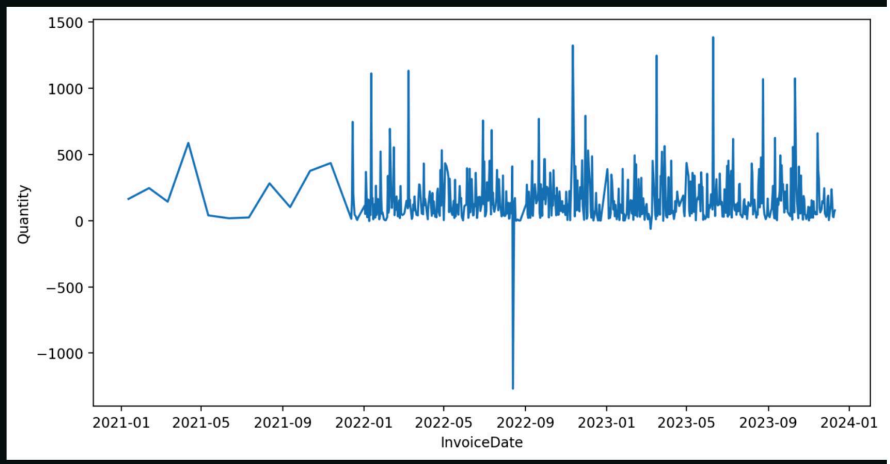
Sales over time for product 84077



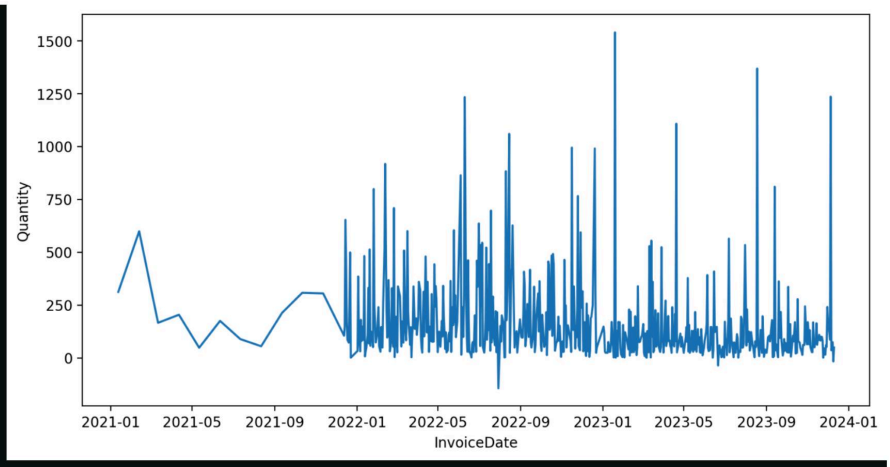
Sales over time for product 85123A

0

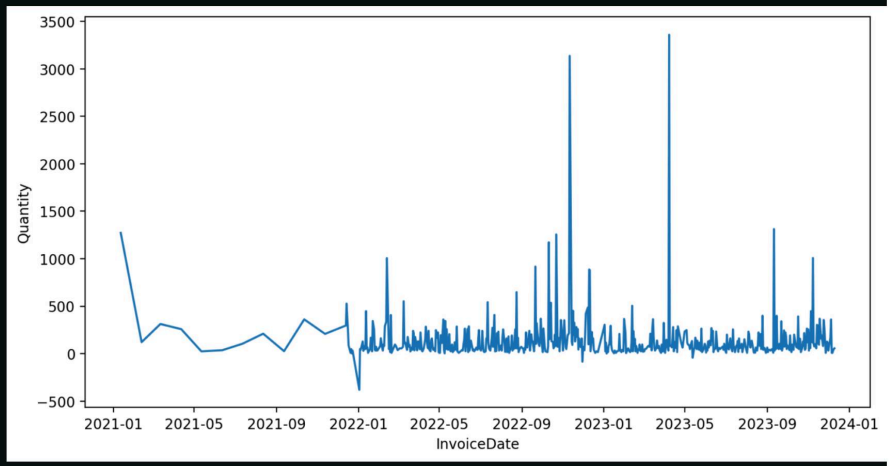
Sales over time for product 85099B



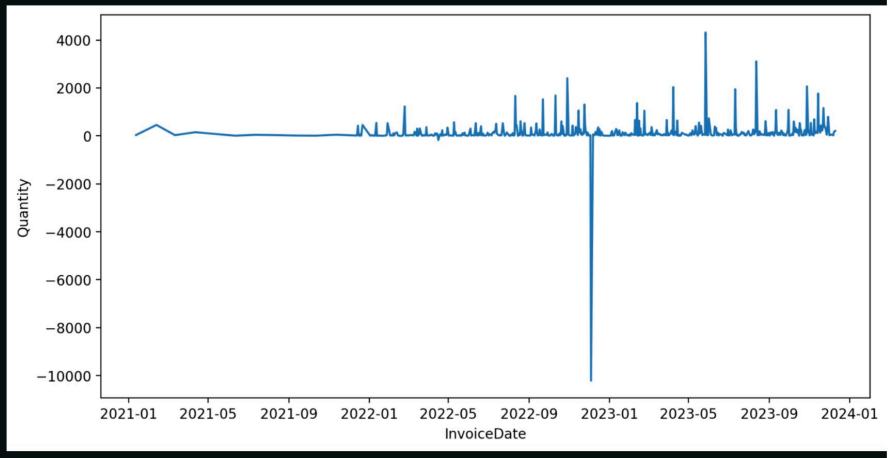
Sales over time for product 21212



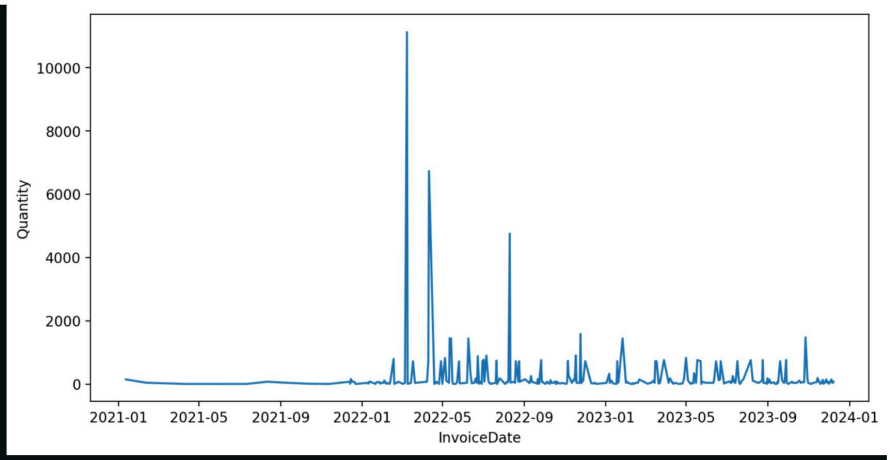
Sales over time for product 84879



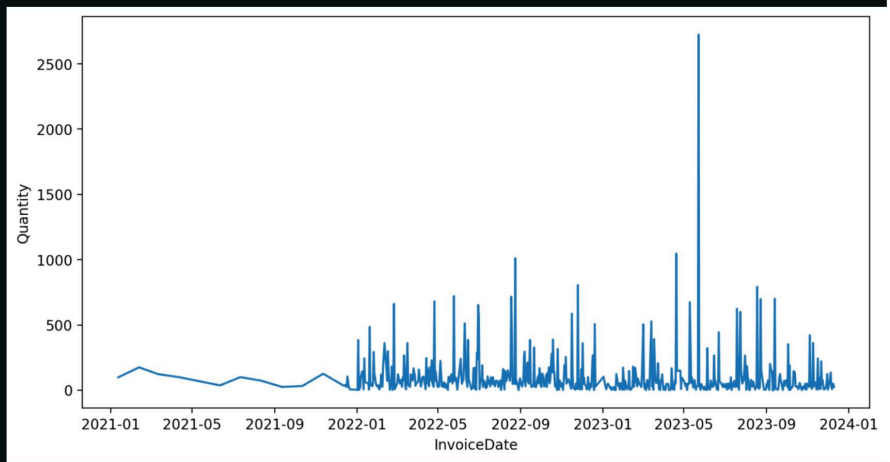
Sales over time for product 22197



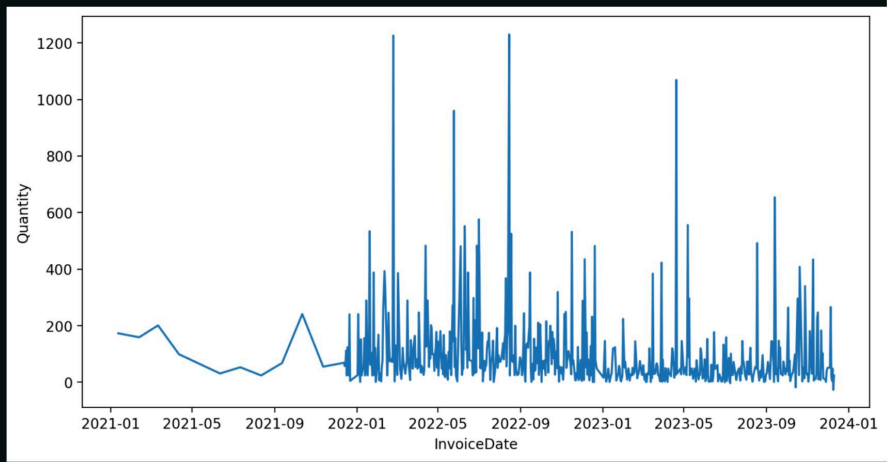
Sales over time for product 17003



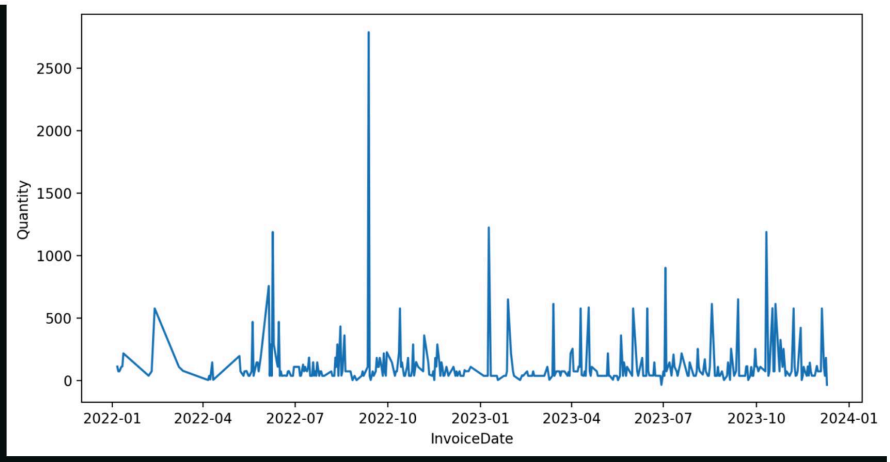
Sales over time for product 21977



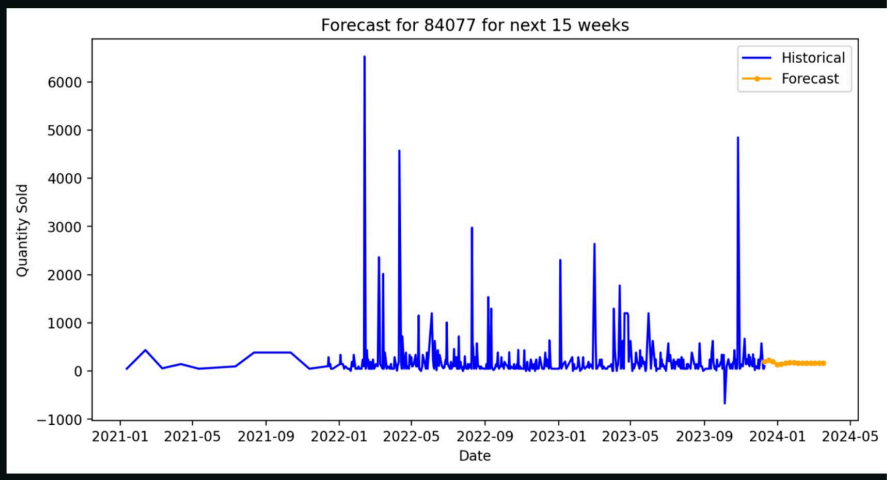
Sales over time for product 84991

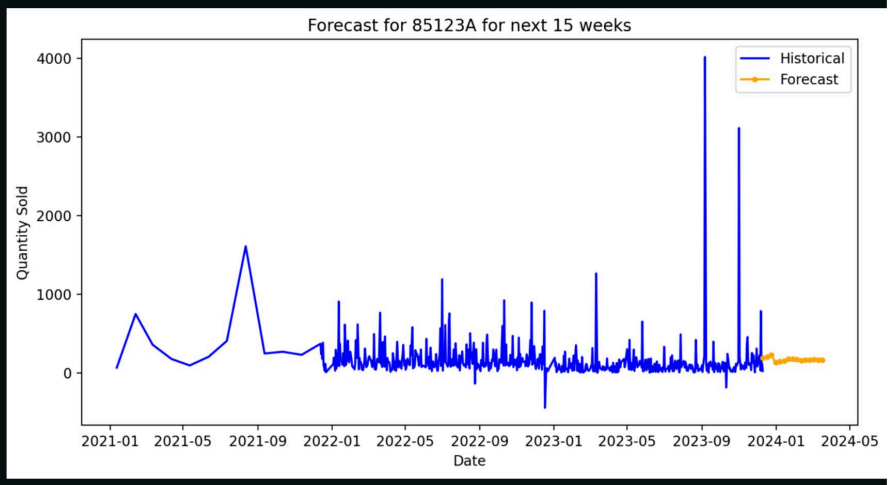


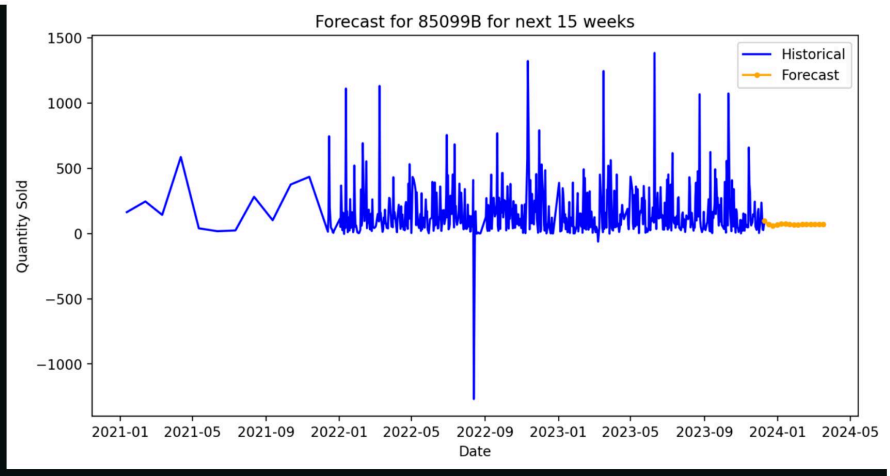
Sales over time for product 22492

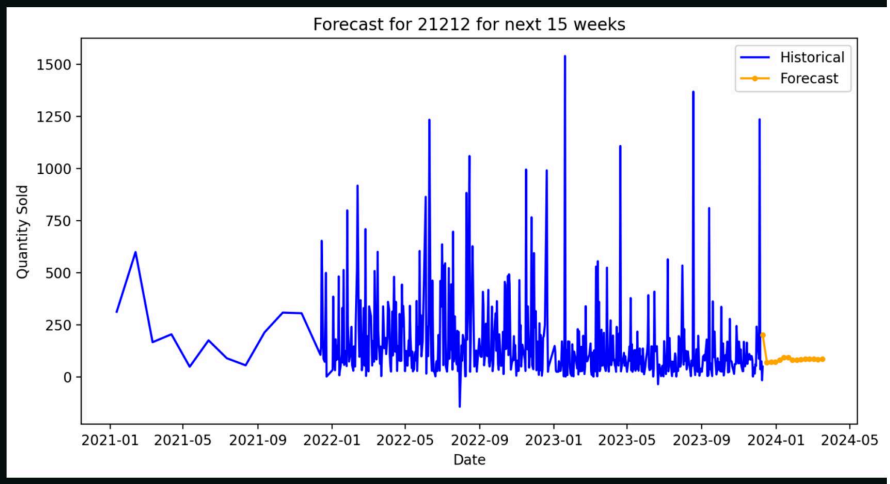


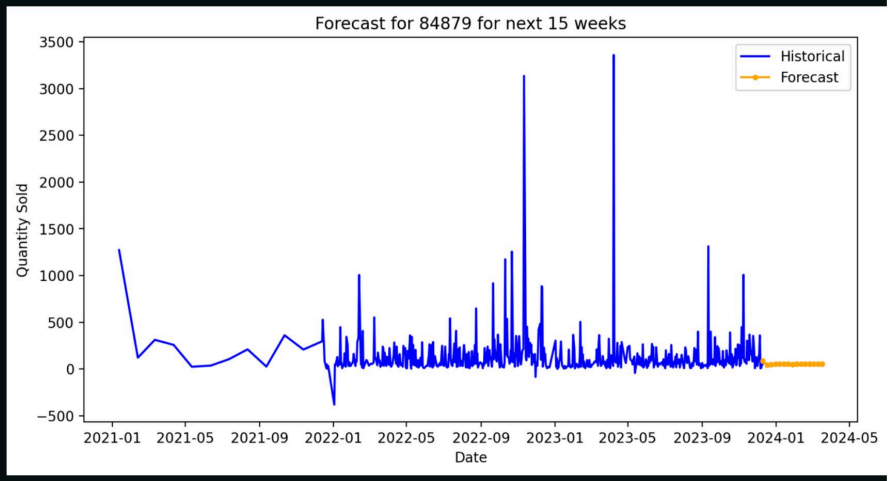
Forecasting Results











Model Evaluation

Download Forecast

Download forecast for 84077

Download forecast for 85123A

Download forecast for 85099B

Download forecast for 21212


Download forecast for 84879