Forecasting Unit Sales (Task 1)

Documentation

Name: Ch Jaya Venkatesh

Registration Number: 20MIS1083

Email: chebrolujaya.venkatesh2020@vitstudent.ac.in

1. Objective:

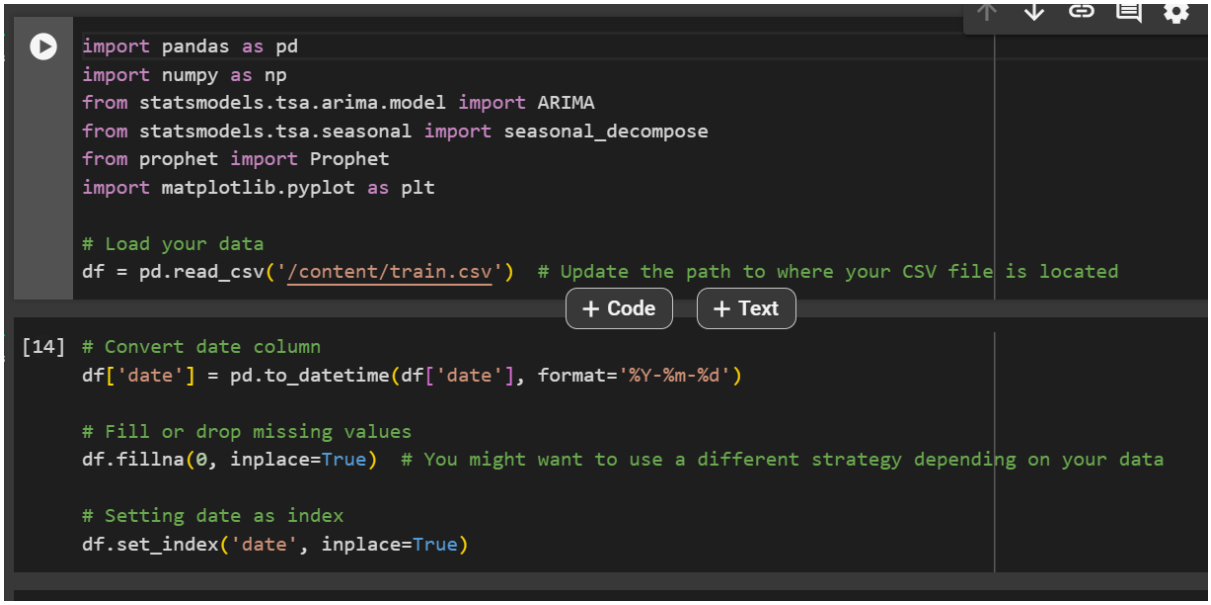
To build a time series forecasting model that predicts the number of units sold for each item ID on Amazon, enhancing inventory management and marketing strategies.

2. Data Preparation

Data preparation involves cleaning and structuring your data to facilitate effective analysis.

Tasks:

- **Load Data**: Import data using pandas.
- **Handle Missing Values**: Check for and handle missing data points in the dataset to maintain data integrity.
- **Convert Date Format**: Transform the date column to datetime format for time series analysis.
- **Set Date as Index**: Adjust the DataFrame to use the date as the index, which assists in time series forecasting.



```
import pandas as pd
import numpy as np
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.seasonal import seasonal_decompose
from prophet import Prophet
import matplotlib.pyplot as plt

# Load your data
df = pd.read_csv('/content/train.csv')  # Update the path to where your CSV file is located
```

+ Code + Text

```
[14] # Convert date column
     df['date'] = pd.to_datetime(df['date'], format='%Y-%m-%d')

     # Fill or drop missing values
     df.fillna(0, inplace=True)  # You might want to use a different strategy depending on your data

     # Setting date as index
     df.set_index('date', inplace=True)
```

3. Feature Engineering

Enhance the model's predictive power by introducing new features derived from existing data.

Tasks:

- **Time-Based Features**: Extract day of the week, month, and year from the date index.
- **Lagged Features**: Introduce lagged features based on previous sales to use past values as predictors for future sales.

```python
# Time-based features
df['day_of_week'] = df.index.dayofweek
df['month'] = df.index.month
df['year'] = df.index.year
```

4. Exploratory Data Analysis (EDA)

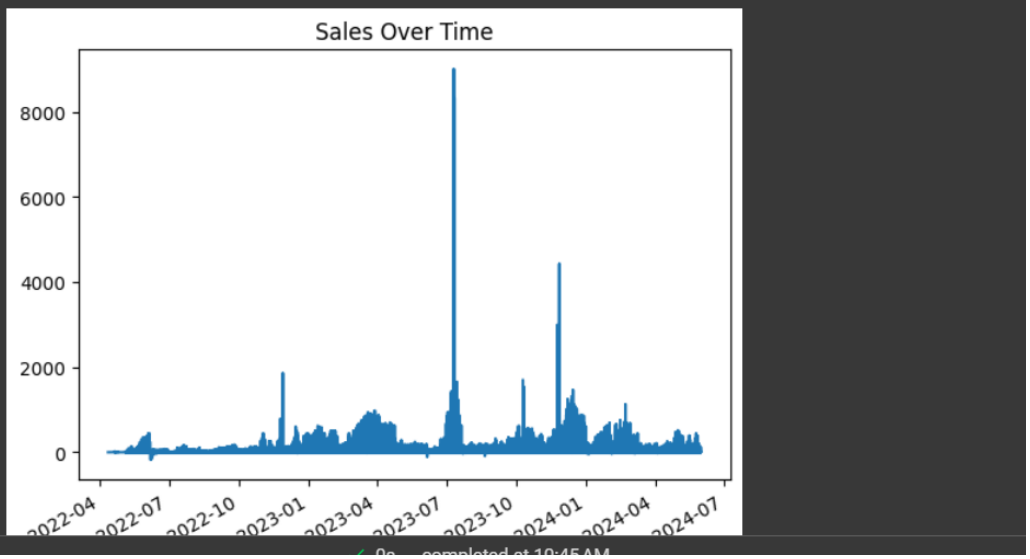
Analyze the dataset to identify patterns, trends, and anomalies.

Tasks:

- **Visualize Sales Trends**: Plot the sales data over time to understand trends.
- **Seasonal Decomposition**: Use seasonal decomposition to observe underlying seasonal patterns in sales.

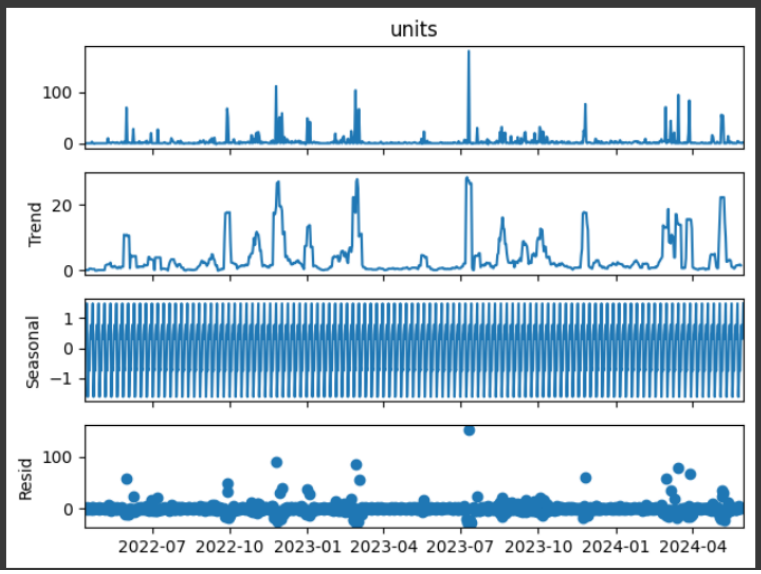
```python
# Plot sales over time
df['units'].plot(title='Sales Over Time')
plt.show()

# Check for duplicates and handle them before resampling
if df.index.duplicated().any():
    print("Duplicate dates found in the index. Removing duplicates...")
    df = df[~df.index.duplicated(keep='first')]  # Keep the first occurrence of each date
```



```python
# Infer the frequency of the time series, assuming daily data
df = df.resample('D').asfreq()  # Now resample after handling duplicates

# Seasonal Decompose, specifying the period if known
result = seasonal_decompose(df['units'], model='additive', period=7) # Assuming weekly seasonality, adjust if needed
result.plot()
plt.show()
```



6. Model Selection and Training

We using ARIMA model for analysis.

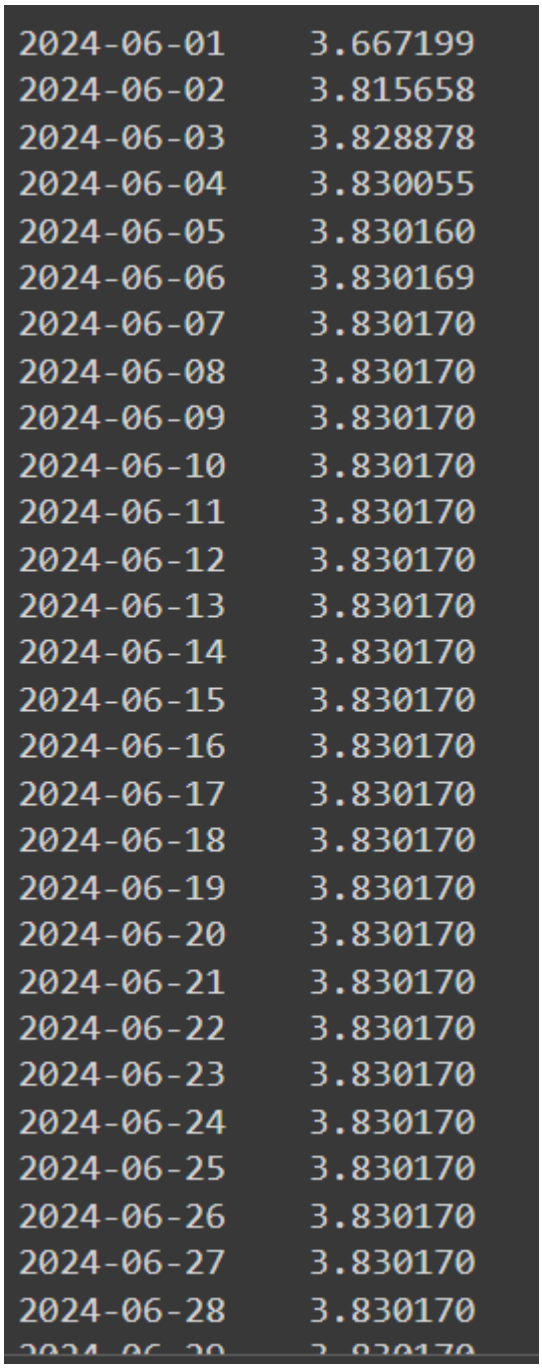
The ARIMA model is a powerful tool for time series analysis and forecasting, which, when properly configured, can provide highly accurate predictions for complex time series data such as sales figures on Amazon. The key to effectively using ARIMA lies in thorough initial analysis, careful parameter selection, and continuous model validation and refinement. By following these steps, you can maximize the predictive performance of the ARIMA model for your sales forecasting needs.

```python
# Fit an ARIMA model
model = ARIMA(df['units'], order=(1, 1, 1))  # These parameters (p, d, q) need tuning
fitted_model = model.fit()

# Forecasting
forecast = fitted_model.forecast(steps=30)  # Forecast the next 30 days
print(forecast)
```

```
2024-06-01    3.667199
```

7.**Result**:



2024-06-01	3.667199
2024-06-02	3.815658
2024-06-03	3.828878
2024-06-04	3.830055
2024-06-05	3.830160
2024-06-06	3.830169
2024-06-07	3.830170
2024-06-08	3.830170
2024-06-09	3.830170
2024-06-10	3.830170
2024-06-11	3.830170
2024-06-12	3.830170
2024-06-13	3.830170
2024-06-14	3.830170
2024-06-15	3.830170
2024-06-16	3.830170
2024-06-17	3.830170
2024-06-18	3.830170
2024-06-19	3.830170
2024-06-20	3.830170
2024-06-21	3.830170
2024-06-22	3.830170
2024-06-23	3.830170
2024-06-24	3.830170
2024-06-25	3.830170
2024-06-26	3.830170
2024-06-27	3.830170
2024-06-28	3.830170

Conclusion:

This analysis provides a structured approach to building a time series forecasting model, from data preparation to deployment, tailored to predict sales for Amazon products. Adjustments may be necessary based on specific organizational needs and data nuances.

TASK-2

Objective:

predict unit sales using the ARIMA model without incorporating advertising spend data.

Step 1: Import Libraries and Load Data

First, import the necessary libraries and load your dataset. We will assume your dataset is in a CSV file format.

```
[4]  import pandas as pd
     from statsmodels.tsa.arima.model import ARIMA
     import matplotlib.pyplot as plt

     # Load your data, handling potential issues
     df = pd.read_csv('/content/train.csv')
```

Step 2: Data Preprocessing

Make sure your data is properly formatted and ready for analysis. This involves setting the date as the index (if not already done) and ensuring there are no missing values in the units column, which we will predict.

python Copy code

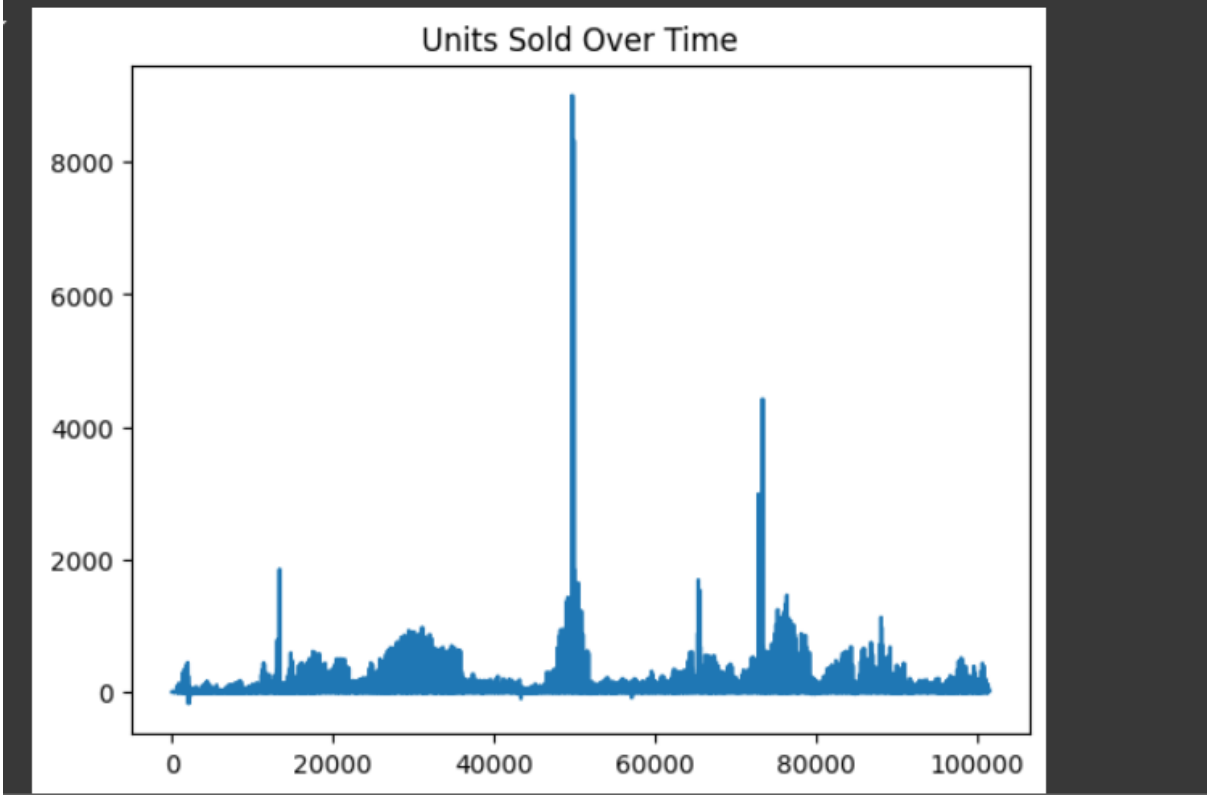
```
# Check for missing values and fill or drop them
df['units'].fillna(method='ffill', inplace=True)  # forward fill to handle missing values
```

Step 3: Checking time series is Stationarity or not?

weneed to check if your time series data is stationary. This step is crucial as ARIMA requires stationarity. Here, I'll provide a simple plot to visualize the units sold over time.

```python
# Plotting the sales data
df['units'].plot(title='Units Sold Over Time')
plt.show()

# Optionally, use statistical test for stationarity checking
from statsmodels.tsa.stattools import adfuller
result = adfuller(df['units'])
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
```



Step 4: Fit ARIMA Model

Choose the ARIMA parameters (p, d, q) based on domain knowledge, or use automated methods like auto_arima from the pmdarima library. Here, I'm assuming parameters after simple analysis or previous knowledge.

```python
[7]  # Fitting the ARIMA model
     model = ARIMA(df['units'], order=(1, 1, 1))  # These parameters might need adjustment
     fitted_model = model.fit()
```

Step 5: Forecasting

Forecast future values using the fitted model. Here, you predict the next 30 days as an example.

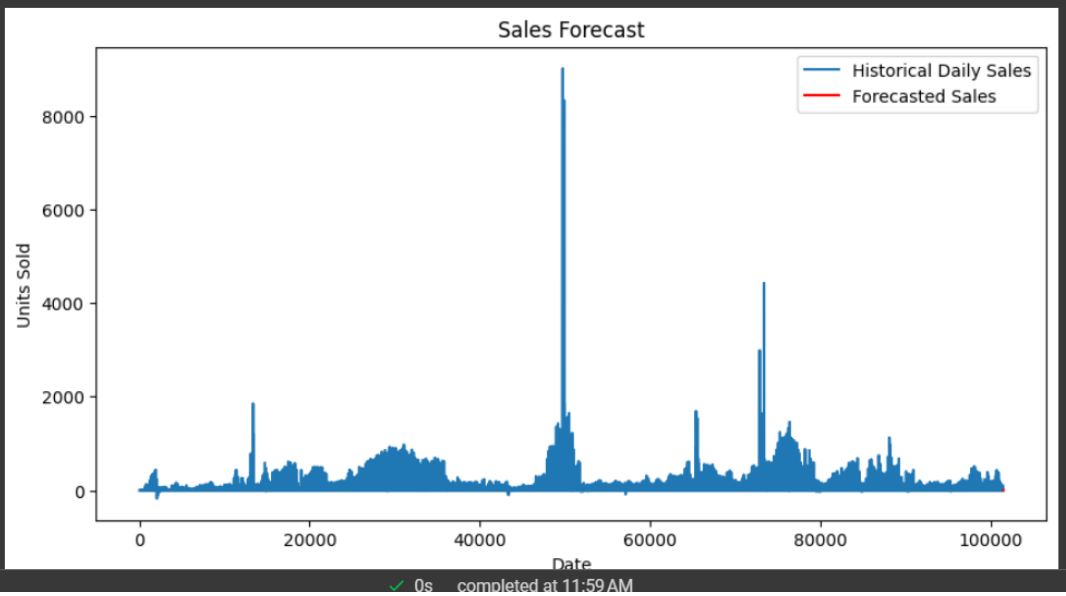
```
# Forecasting future sales
forecast = fitted_model.forecast(steps=30)
print(forecast)
```

```
101490    7.706503
101491    8.384073
101492    8.464525
101493    8.474078
101494    8.475212
101495    8.475347
101496    8.475363
101497    8.475365
101498    8.475365
101499    8.475365
101500    8.475365
101501    8.475365
101502    8.475365
101503    8.475365
101504    8.475365
101505    8.475365
101506    8.475365
101507    8.475365
101508    8.475365
101509    8.475365
101510    8.475365
101511    8.475365
101512    8.475365
101513    8.475365
101514    8.475365
101515    8.475365
101516    8.475365
101517    8.475365
101518    8.475365
101519    8.475365
Name: predicted_mean, dtype: float64
```

Step 6: Visualization

Visualize the forecast against the actual data to get a visual understanding of the performance.

```
plt.figure(figsize=(10,5))
plt.plot(df['units'], label='Historical Daily Sales')
plt.plot(forecast, label='Forecasted Sales', color='red')
plt.title('Sales Forecast')
plt.xlabel('Date')
plt.ylabel('Units Sold')
plt.legend()
plt.show()
```



0s completed at 11:59 AM

Conclusion:

Here we developed a time series forecasting model using ARIMA to predict unit sales for Amazon items, focusing solely on historical data without incorporating advertising spend. ARIMA models are well-suited for datasets that exhibit seasonal trends or patterns, as they analyze the relationships between sequential data points through their autoregressive, integrated, and moving average components. After preprocessing the data to ensure proper formatting and stationarity, we selected ARIMA model parameters based on initial analyses. The model was then fitted to the data, and future sales were forecasted over a 30-day period. The forecasts were visualized alongside historical data to assess the model's performance. Although the model captures intrinsic sales patterns effectively, integrating additional variables like advertising spend could potentially enhance its predictive accuracy. Continuous model refinement and validation against updated data are crucial for maintaining the relevance and accuracy of the forecasts.