Sales Forecasting Using Time Series

1. Introduction

This project aims to develop a model for accurately forecasting future sales using time series analysis. We will explore historical sales data, perform exploratory data analysis (EDA), and apply forecasting models like ARIMA and Prophet to predict future sales trends.

2. Scope

The scope of this project includes:

- - Understanding historical sales trends.
- - Identifying seasonality and anomalies in the data.
- - Using statistical and machine learning models for forecasting.
- - Evaluating model performance using error metrics.

3. Data Collection

We are using publicly available historical sales data. The dataset can be accessed from Kaggle.

4. Step-by-Step Analysis

Step 1: Import Required Libraries

First, install and import the necessary Python libraries:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.arima.model import ARIMA
from prophet import Prophet
from sklearn.metrics import mean_absolute_error, mean_squared_error
```

Step 2: Load the Data

Load the dataset and perform initial data checks:

```
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/shampoo.csv"
df = pd.read_csv(url, header=0, parse_dates=[0], index_col=0)
df.columns = ["Sales"]
df.index.name = "Date"
```

```
print(df.head())
```

Step 3: Data Preprocessing

Check for missing values, handle duplicates, and format the date column:

```
print(df.isnull().sum())
print(df.duplicated().sum())
df.index = pd.to_datetime(df.index, format='%m-%d')
print(df.describe())
```

Step 4: Exploratory Data Analysis (EDA)

Visualizing sales trends and decomposing time series data:

```
plt.figure(figsize=(12, 6))
plt.plot(df, marker='o', linestyle='-')
plt.title("Monthly Sales Over Time")
plt.xlabel("Date")
plt.ylabel("Sales")
plt.grid()
plt.show()

decomposition = seasonal_decompose(df, model="multiplicative", period=12)
decomposition.plot()
plt.show()
```

Step 5: Time Series Forecasting Models

ARIMA Model

Applying the ARIMA model for forecasting:

```
model = ARIMA(df, order=(5,1,0))
model_fit = model.fit()
df["ARIMA_Forecast"] = model_fit.predict(start=len(df), end=len(df)+12, dynamic=False)
plt.figure(figsize=(12, 6))
plt.plot(df["Sales"], label="Actual Sales")
plt.plot(df["ARIMA_Forecast"], label="ARIMA Forecast", linestyle="dashed")
plt.title("ARIMA Sales Forecast")
plt.legend()
```

```
plt.show()
```

Prophet Model

Using Facebook's Prophet model for forecasting:

```
prophet_df = df.reset_index().rename(columns={"Date": "ds", "Sales": "y"})
model = Prophet()
model.fit(prophet_df)
future = model.make_future_dataframe(periods=12, freq='M')
forecast = model.predict(future)
model.plot(forecast)
plt.title("Prophet Sales Forecast")
plt.show()
```

Step 6: Model Evaluation

Evaluating the model using MAE and RMSE:

```
actual = df["Sales"][-12:]
arima_forecast = df["ARIMA_Forecast"][-12:]
mae = mean_absolute_error(actual, arima_forecast)
rmse = np.sqrt(mean_squared_error(actual, arima_forecast))
print(f"MAE: {mae}")
print(f"RMSE: {rmse}")
```

Step 7: Conclusion

The project successfully forecasted sales using ARIMA and Prophet models. The model with the lower error is preferred for future forecasting. The sales forecasting model provides valuable insights into sales trends. It enables businesses to plan for peak and low-demand periods, optimize inventory, and improve resource allocation. Further improvements can include incorporating external factors such as promotions and holidays.