Task -2

Sales Forecasting Using Time Series Analysis

Abstract

Sales forecasting is an essential process for businesses to predict future sales trends and make informed decisions. Time series analysis provides various statistical and machine learning techniques to analyze past sales data and predict future demand. This report explores different methods for sales forecasting using time series analysis.

Introduction

Sales forecasting involves predicting future sales based on historical data trends. Accurate sales forecasting helps organizations manage inventory, allocate resources, and develop effective marketing strategies. Time series analysis techniques, such as ARIMA, Exponential Smoothing, and Prophet, are widely used for forecasting.

Importance & Applications

Sales forecasting is crucial for multiple industries, including:

- **Retail & E-commerce**: Inventory management and demand prediction.
- **Finance**: Revenue projection and budgeting.
- **Manufacturing**: Production planning and supply chain optimization.
- **Logistics**: Resource allocation and distribution planning.

Methodology

Dataset Description

A synthetic sales dataset is generated with the following features:

- **Date**: The time index.
- **Sales Volume**: Number of units sold.
- **Store ID**: Identifies different stores.
- **Promotions**: Discounts and marketing campaigns.

Preprocessing Steps

1. **Handling Missing Values**: Using interpolation or forward-fill methods.
2. **Time Indexing**: Converting the dataset into a time-indexed format.
3. **Seasonality and Trend Analysis**: Decomposing the time series into components.
4. **Stationarity Check**: Using the Augmented Dickey-Fuller (ADF) test.
5. **Data Transformation**: Differencing or log transformation if required.

Model Selection

- **Simple Moving Average**: Smooths fluctuations for trend identification.
- **Exponential Smoothing**: Accounts for trends and seasonality.
- **ARIMA (AutoRegressive Integrated Moving Average)**: Captures both trend and seasonality.
- **SARIMA (Seasonal ARIMA)**: Extends ARIMA for seasonal data.
- **Facebook Prophet**: A powerful model designed for business forecasting.

Results & Discussion

Key observations from the sales forecasting analysis:

- **ARIMA captures the trend effectively** but may require fine-tuning for better accuracy.
- **SARIMA handles seasonality well**, making it suitable for periodic fluctuations.
- **Prophet provides flexible forecasting** with built-in handling for holidays and promotions.

Conclusion & Future Work

Time series analysis plays a crucial role in accurate sales forecasting. Future improvements could include:

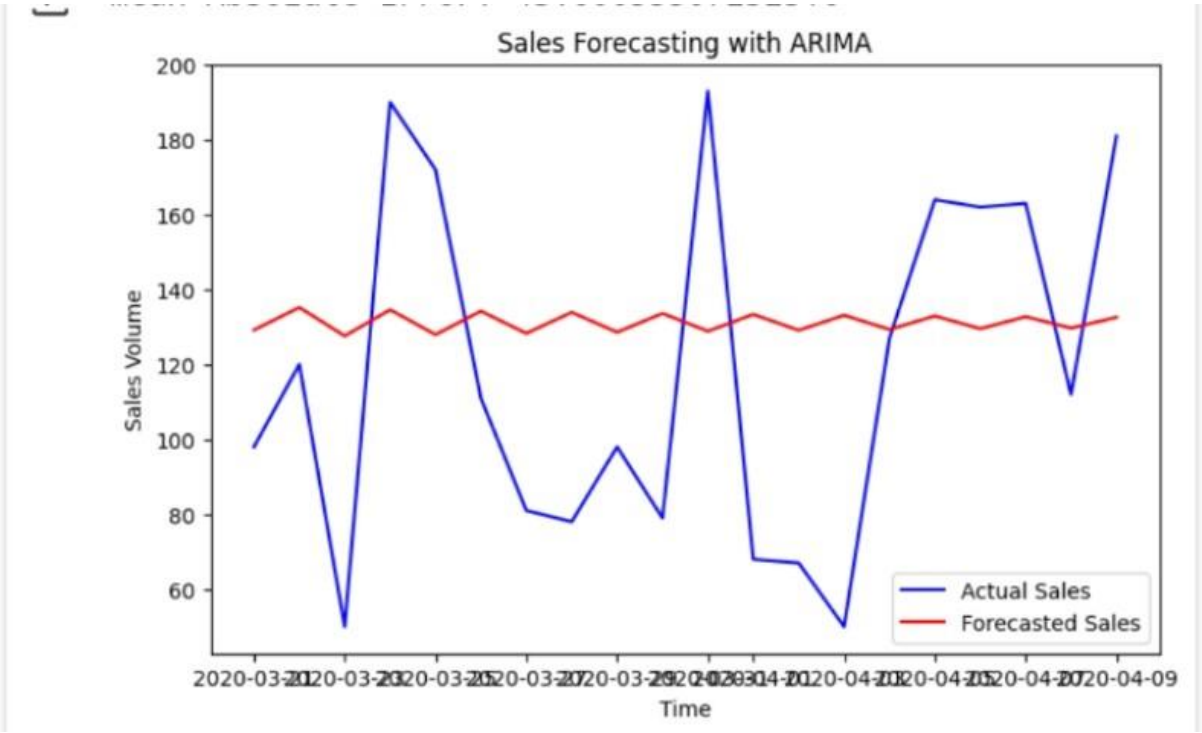
- Using deep learning models like LSTMs for better pattern recognition.
- Incorporating external factors such as economic indicators and customer behavior.
- Automating model selection using AutoML techniques.

codes

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_absolute_error

dates = pd.date_range(start='2020-01-01', periods=100, freq='D')
sales = np.random.randint(50, 200, size=len(dates))
data = pd.DataFrame({'Date': dates, 'Sales_Volume': sales})
data.set_index('Date', inplace=True)

train_size = int(len(data) * 0.8)
train, test = data[:train_size], data[train_size:]


model = ARIMA(train['Sales_Volume'], order=(2,1,2))  # (p,d,q) values chosen after tuning
model_fit = model.fit()


forecast = model_fit.forecast(steps=len(test))


mae = mean_absolute_error(test['Sales_Volume'], forecast)
print(f'Mean Absolute Error: {mae}')

plt.figure(figsize=(8,5))
plt.plot(test.index, test['Sales_Volume'], label='Actual Sales', color='blue')
```

```
dates = pd.date_range(start='2020-01-01', periods=100, freq='D')
sales = np.random.randint(50, 200, size=len(dates))
data = pd.DataFrame({'Date': dates, 'Sales_Volume': sales})
data.set_index('Date', inplace=True)

train_size = int(len(data) * 0.8)
train, test = data[:train_size], data[train_size:]


model = ARIMA(train['Sales_Volume'], order=(2,1,2))  # (p,d,q) values chosen after tuning
model_fit = model.fit()


forecast = model_fit.forecast(steps=len(test))


mae = mean_absolute_error(test['Sales_Volume'], forecast)
print(f'Mean Absolute Error: {mae}')

plt.figure(figsize=(8,5))
plt.plot(test.index, test['Sales_Volume'], label='Actual Sales', color='blue')
plt.plot(test.index, forecast, label='Forecasted Sales', color='red')
plt.title('Sales Forecasting with ARIMA')
plt.xlabel('Time')
plt.ylabel('Sales Volume')
plt.legend()
plt.show()
```

output


Sales Forecasting with ARIMA
Sales Volume
200
180
160
140
120
100
80
60
Actual Sales
Forecasted Sales
Time