Python Project

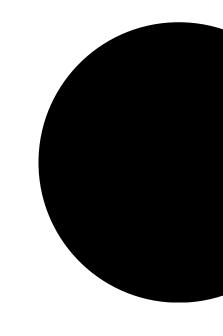


FOR DATA ANALYST

Time Series Analysis

linkedin.com/in/ileonjose





Sales Forecasting Using Time Series Analysis

linkedin.com/in/ileonjose



Overview: Predict future sales using historical sales data and time series analysis techniques. This project showcases your analytical skills, ability to work with time-dependent data, and proficiency in statistical modeling.

Dataset:

https://www.kaggle.com/competitions/store-sales-time-series-forecasting/data



Step 1: Gather Data

Use a public sales dataset or create a synthetic dataset for this project. You can use libraries like Pandas to handle your data.

1. Data Source:

 You can find datasets on platforms like Kaggle, or create a simple CSV file with columns for date and sales.



```
import pandas as pd

# Load the dataset
data = pd.read_csv('sales_data.csv', parse_dates=['date'])
data.set_index('date', inplace=True)
```



Step 2: Preprocess Data

Conduct exploratory data analysis (EDA) to understand your dataset and preprocess it.

- 1. Visualize Sales Trends:
- Plot historical sales data to identify trends, seasonality, or anomalies.



```
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))
plt.plot(data.index, data['sales'], label='Sales')
plt.title('Historical Sales Data')
plt.xlabel('Date')
plt.ylabel('Sales')
plt.legend()
plt.show()
```



Handle Missing Values:

 Fill or interpolate any missing values for continuity.

```
Python

data['sales'].fillna(method='ffill', inplace=True) # Forward fill
```



Step 3: Decompose the Time Series

Decompose the time series into trend, seasonality, and residuals to better understand the components.

- 1. Use statsmodels for Decomposition:
- Break down the series using seasonal decomposition.

```
from statsmodels.tsa.seasonal import seasonal_decompose

decomposed = seasonal_decompose(data['sales'], model='additive')
  decomposed.plot()
  plt.show()
```



Step 4: Build a Forecasting Model

Implement a forecasting model such as ARIMA (AutoRegressive Integrated Moving Average) to predict future sales.

1. Define the Model:

 Select parameters based on ACF/PACF plots or use auto-ARIMA for automated selection.



```
from pmdarima import auto_arima

model = auto_arima(data['sales'], seasonal=True, m=12) #
Monthly seasonality
```

2. Fit the Model:

• Fit the ARIMA model to your data.

```
model.fit(data['sales'])
```



Step 5: Make Predictions and Visualize

Forecast future sales and visualize the results against historical data.

- 1. Generate Forecasts:
- Use the model to predict future sales.

```
python

future_dates = pd.date_range(start='2024-01-01', periods=12, freq='M')

# Predict for next year
forecast = model.predict(n_periods=12)
```



2. Plot the Forecast:

```
plt.figure(figsize=(12, 6))
plt.plot(data.index, data['sales'], label='Historical Sales')
plt.plot(future_dates, forecast, label='Forecasted Sales',
color='orange')
plt.title('Sales Forecasting')
plt.xlabel('Date')
plt.ylabel('Sales')
plt.legend()
plt.show()
```



Step 6: Interpret Insights

Compile a report summarizing your findings:

- Accuracy Assessment: Discuss the model's accuracy using metrics like RMSE (Root Mean Squared Error).
- Sales Trends: Analyze how forecasted sales can inform business decisions.
- Seasonal Trends: Highlight any seasonal patterns that may affect sales strategy.



Recruiters POV

- This project emphasizes your ability to handle time series data, implement statistical modeling, and draw actionable insights.
- It showcases skills in data visualization and the ability to communicate complex findings clearly.





Found this helpful? Repost!

linkedin.com/in/ileonjose