**S REVENUE FORCASTING**

**Objectives:**

Forecasting revenue for a telecom (telco) company using time series forecasting involves analyzing historical revenue data to make predictions about future revenue trends.

**Specific**:

* Implement EDA
* Develop and
* Evaluate Visualization

**Methods**:

1. Describe The Dataset *(source descriptive analytics, etc.)*
   1. The data is organized as a time series, with a record for each day representing the revenue for that day.
   2. The dataset contains two columns (day and revenue) and 731 rows.
   3. This starts by exploring basic statistics of the dataset, such as mean, median, standard deviation, etc., for numerical values. This will give an initial understanding about the data.
2. Process, Techniques, and Methods for Objective

To forecast trends and capture potential seasonality in the telco revenue data, you can enhance your time series forecasting approach. One way to achieve this is by using the Seasonal-Trend decomposition using LOESS (STL) decomposition, which decomposes the time series into three components: Seasonal, Trend, and Residual.

Here's an extended version of the previous example incorporating STL decomposition using the statsmodels library:

Step 1: Import Libraries

python

Copy code

import pandas as pd

import matplotlib.pyplot as plt

from statsmodels.tsa.seasonal import STL

Step 2: Load and Explore Data

python

Copy code

data = pd.read\_csv("your\_telecom\_revenue\_data.csv")

data['Date'] = pd.to\_datetime(data['Date'])

data.set\_index('Date', inplace=True)

Step 3: STL Decomposition

Decompose the time series into Seasonal, Trend, and Residual components.

python

Copy code

stl = STL(data['Revenue'], seasonal=13) # Adjust the seasonal parameter based on your data

result = stl.fit()

trend = result.trend

seasonal = result.seasonal

residual = result.resid

# Visualize the decomposition

plt.figure(figsize=(12, 8))

plt.subplot(4, 1, 1)

plt.plot(data['Revenue'], label='Original Data')

plt.legend()

plt.subplot(4, 1, 2)

plt.plot(trend, label='Trend')

plt.legend()

plt.subplot(4, 1, 3)

plt.plot(seasonal, label='Seasonal')

plt.legend()

plt.subplot(4, 1, 4)

plt.plot(residual, label='Residual')

plt.legend()

plt.tight\_layout()

plt.show()

Step 4: Check Stationarity

Check the stationarity of the residual component after decomposition.

python

Copy code

result\_residual = adfuller(residual.dropna())

print('ADF Statistic (Residual):', result\_residual[0])

print('p-value (Residual):', result\_residual[1])

Step 5: Forecasting with ARIMA on Residuals

Fit an ARIMA model to the residual component and make predictions.

python

Copy code

order\_residual = (p, d, q) # Adjust these parameters based on analysis

model\_residual = ARIMA(residual.dropna(), order=order\_residual)

results\_residual = model\_residual.fit()

# Make predictions

forecast\_residual = results\_residual.get\_forecast(steps=forecast\_steps)

forecast\_residual\_index = pd.date\_range(start=data.index[-1] + pd.DateOffset(1), periods=forecast\_steps, freq='M')

forecast\_residual\_series = pd.Series(forecast\_residual.predicted\_mean.values, index=forecast\_residual\_index)

# Visualize the forecasted residuals

plt.figure(figsize=(12, 4))

plt.plot(residual, label='Residual')

plt.plot(forecast\_residual\_series, label='Forecasted Residual', color='red')

plt.legend()

plt.title('Forecasted Residuals')

plt.show()

Step 6: Combine Components

Combine the trend, seasonal, and forecasted residuals to obtain the overall forecast.

python

Copy code

# Forecasted values

forecast\_combined = trend + seasonal + forecast\_residual\_series

# Plot original data and forecast

plt.figure(figsize=(12, 8))

plt.plot(data['Revenue'], label='Historical Revenue')

plt.plot(forecast\_combined, label='Forecasted Revenue', color='red')

plt.title('Telco Revenue Forecast with STL Decomposition')

plt.xlabel('Date')

plt.ylabel('Revenue')

plt.legend()

plt.show()