

| **TITLE : To perform time series analysis on health care** |
| --- |

**AIM:** To perform forecasting using time series analysis

**Expected OUTCOME of Experiment:**

CO4: Perform Time series Analytics and forecasting

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

Google colab

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Pre Lab/ Prior Concepts:**

Students should have a basic understanding of: Time series Analytics and forecasting

**Procedure:**

**Data set Used: Hospital\_patients\_datasets**

**Step1: Select and Load the dataset**

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from prophet import Prophet

# Load the dataset

file\_path = '/content/Hospital\_patients\_datasets.csv'

data = pd.read\_csv(file\_path)

# Convert 'ScheduledDay' and 'AppointmentDay' to datetime format

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

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

# Create new features: Appointment Weekday, Scheduled Weekday, Days Between Scheduling and Appointment

data['AppointmentWeekday'] = data['AppointmentDay'].dt.day\_name()

data['ScheduledWeekday'] = data['ScheduledDay'].dt.day\_name()

data['DaysBetween'] = (data['AppointmentDay'] - data['ScheduledDay']).dt.days

**# Forecasting Daily Attendance**

**# Step 1: Preprocess the data to aggregate daily attendance**

**data['Attended'] = data['No-show'].apply(lambda x: 0 if x == 'Yes' else 1)**

**attendance\_daily = data.groupby('AppointmentDay')['Attended'].sum().reset\_index()**

**Step2: Convert 'ScheduledDay' and 'AppointmentDay' to datetime format**

# Step 2: Rename columns for Prophet ('ds' for date, 'y' for the target)

attendance\_daily.columns = ['ds', 'y']

# Ensure the 'ds' column is in datetime format and without timezone

attendance\_daily['ds'] = pd.to\_datetime(attendance\_daily['ds']).dt.tz\_localize(None)

# Ensure 'y' is numeric

attendance\_daily['y'] = pd.to\_numeric(attendance\_daily['y'], errors='coerce')

# Drop any rows with missing values (NaNs)

attendance\_daily = attendance\_daily.dropna()

**Step 3: Forecasting Daily Attendance  
# Step 3: Initialize Prophet model for forecasting**

**model = Prophet()**

**Step4: Initialize Prophet model for forecasting**

**# Step 4: Fit the model**

**model.fit(attendance\_daily)**

**Step 5: Fit the model**

# Step 5: Create future dates for prediction (e.g., next 30 days)

future\_dates = model.make\_future\_dataframe(periods=30)

**Step 6: Predict future attendance**

# Step 6: Predict future attendance

forecast = model.predict(future\_dates)

**Step 7: Plot the forecast**

# Step 7: Plot the forecast

fig = model.plot(forecast)

plt.title('Forecast of Daily Appointment Attendance', fontsize=16)

plt.xlabel('Date')

plt.ylabel('Attendance')

plt.show()

**Step 8: Exploratory Data Analysis Functions**

# Exploratory Data Analysis Functions

def plot\_no\_show\_distribution():

"""Plot no-show distribution"""

plt.figure(figsize=(8, 6))

sns.countplot(x='No-show', data=data, palette="Set2")

plt.title('Distribution of No-shows', fontsize=16)

plt.ylabel('Count')

plt.xlabel('No-show Status')

plt.show()

def plot\_age\_distribution():

"""Plot age distribution of patients"""

plt.figure(figsize=(10, 6))

sns.histplot(data=data, x='Age', bins=50, kde=True, color='skyblue')

plt.title('Age Distribution of Patients', fontsize=16)

plt.xlabel('Age')

plt.ylabel('Count')

plt.show()

def plot\_age\_vs\_no\_show():

"""Plot relationship between Age and No-show status"""

plt.figure(figsize=(10, 6))

sns.boxplot(x='No-show', y='Age', data=data, palette="Set3")

plt.title('Age vs. No-show Status', fontsize=16)

plt.xlabel('No-show Status')

plt.ylabel('Age')

plt.show()

def plot\_medical\_conditions\_vs\_no\_show():

"""Plot relationship between medical conditions and No-show status"""

plt.figure(figsize=(12, 6))

medical\_conditions = ['Hipertension', 'Diabetes', 'Alcoholism', 'Handcap']

for condition in medical\_conditions:

plt.figure()

sns.countplot(x='No-show', hue=condition, data=data, palette="Set2")

plt.title(f'Relationship between {condition} and No-show')

plt.ylabel('Count')

plt.xlabel('No-show Status')

plt.legend(title=condition)

plt.show()

def plot\_sms\_vs\_no\_show():

"""Plot relationship between SMS reminders and No-show status"""

plt.figure(figsize=(10, 6))

sns.countplot(x='No-show', hue='SMS\_received', data=data, palette="Set2")

plt.title('Relationship between SMS Received and No-show', fontsize=16)

plt.ylabel('Count')

plt.xlabel('No-show Status')

plt.legend(title='SMS Received', loc='upper right')

plt.show()

**Step 9: Running the analysis functions**

# Running the analysis functions

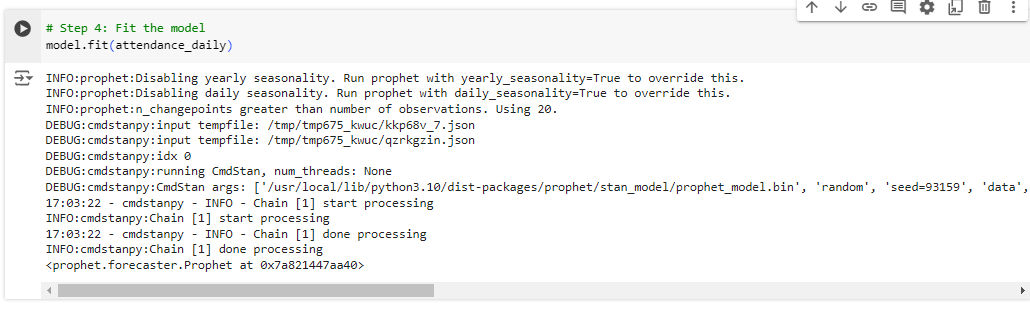
plot\_no\_show\_distribution()

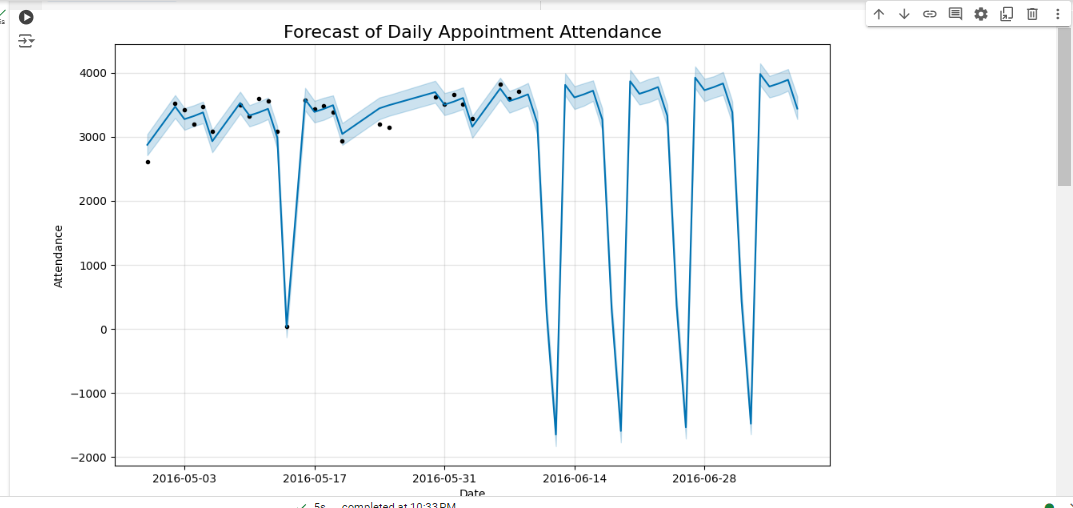
plot\_age\_distribution()

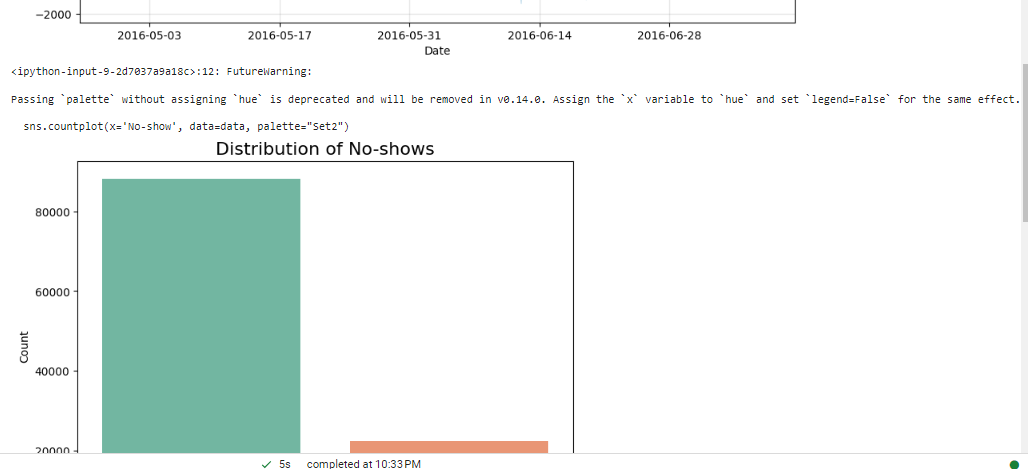
plot\_age\_vs\_no\_show()

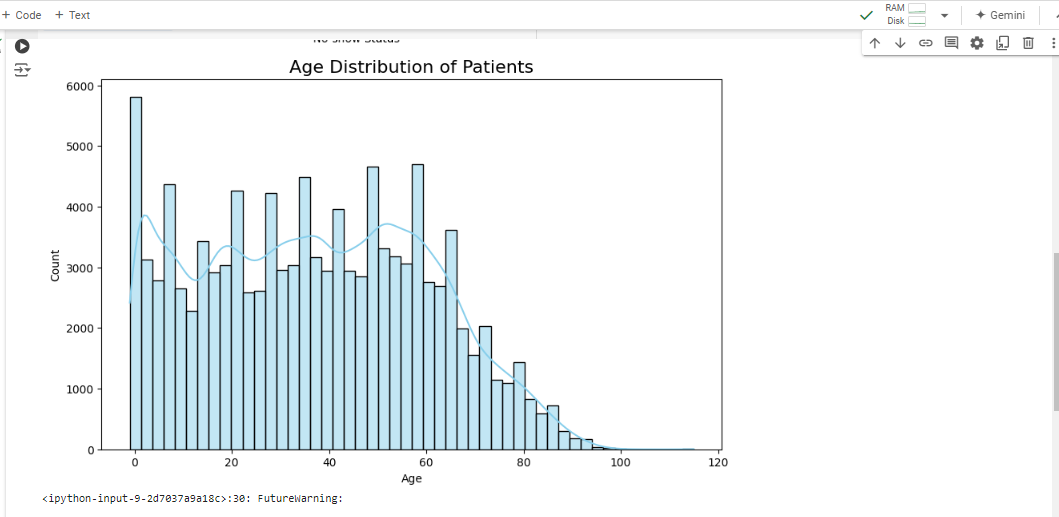
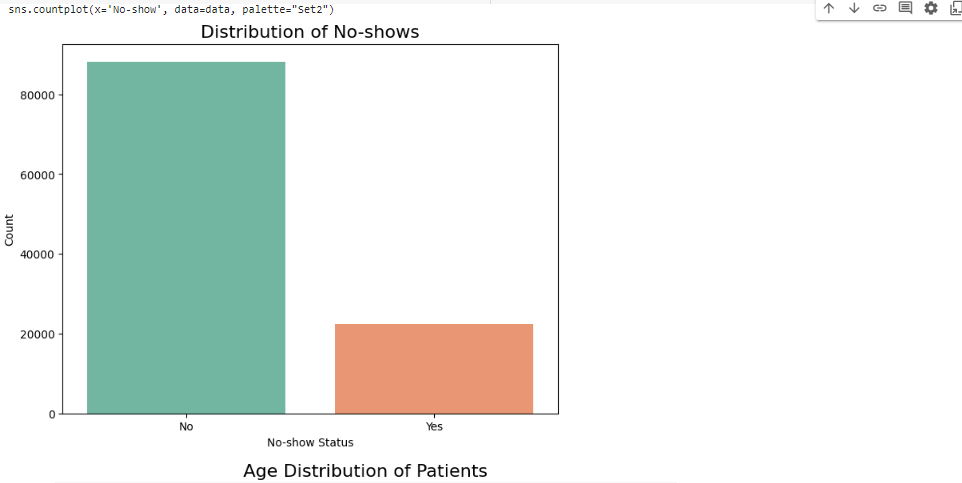
plot\_medical\_conditions\_vs\_no\_show()

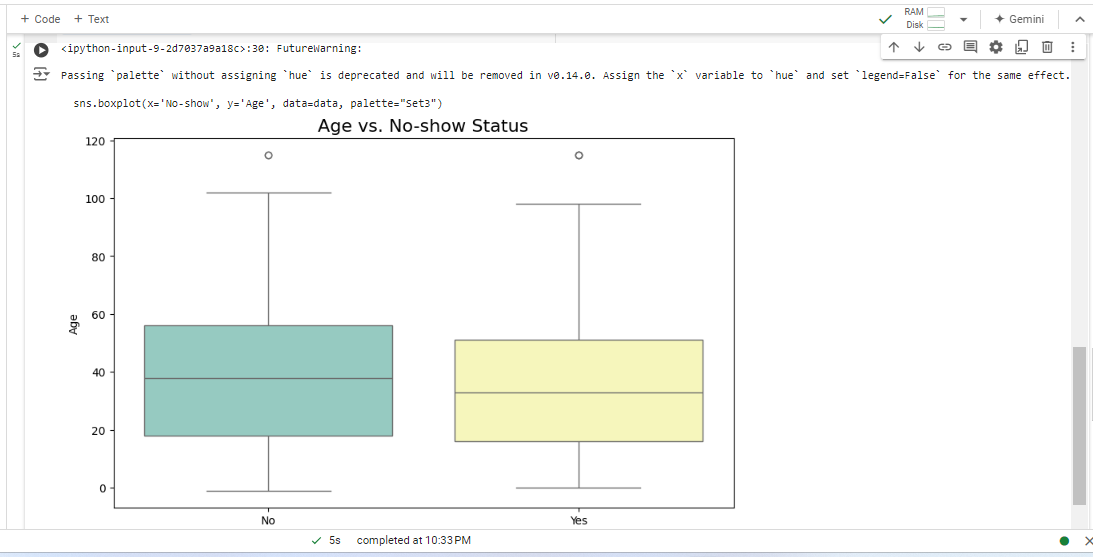
plot\_sms\_vs\_no\_show()

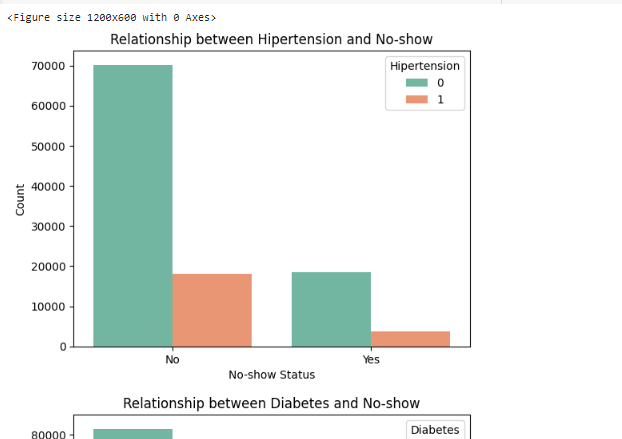
****

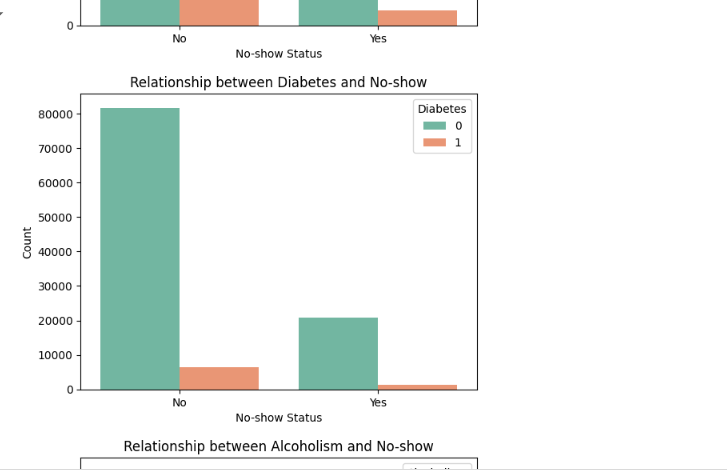
****

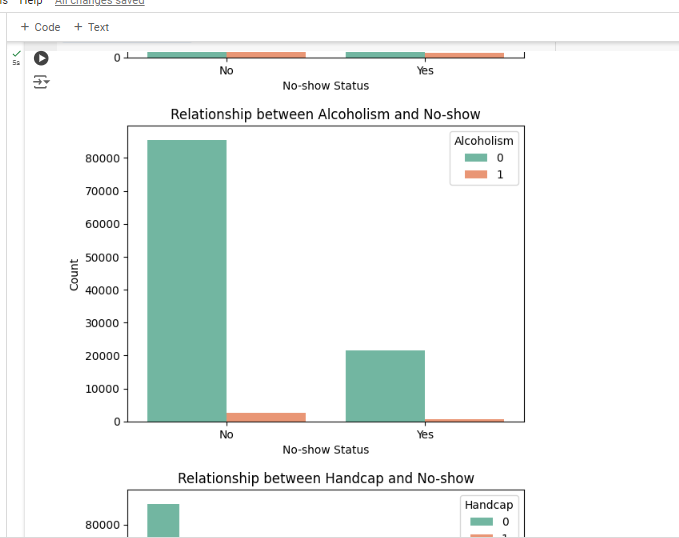
****

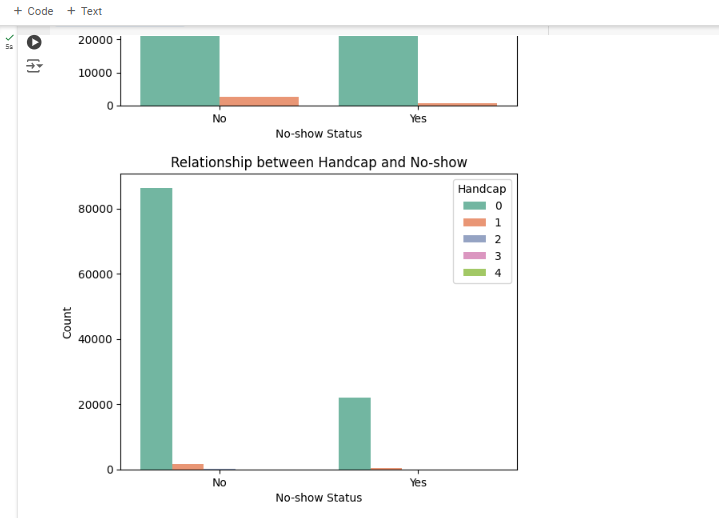
****

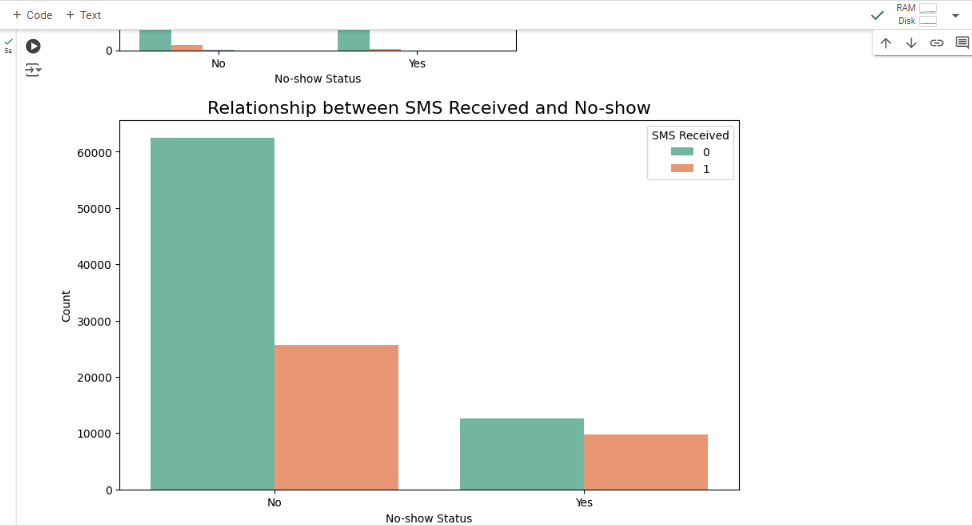
****

****

****

****

****

****

**Date: 17/10/24 Signature of faculty in-charge**

**Post Lab Descriptive Questions:**

1. Explain the components of time series?

A time series typically consists of four main components:

* **Trend**: Long-term movement indicating the overall direction (increasing, decreasing, or constant).
* **Seasonality**: Regular, periodic fluctuations at specific intervals, influenced by external factors (e.g., holidays).
* **Cyclic Patterns**: Irregular fluctuations driven by economic factors, reflecting long-term cycles lasting several years.
* **Irregular (Noise)**: Random variations or outliers not explained by trend, seasonality, or cyclic behavior.

1. How do you handle seasonality in time series data? What methods or transformations can you apply?

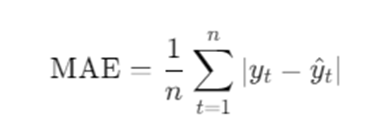
There are several methods to address seasonality in time series data:

* **Seasonal Decomposition**: Breaks down time series into trend, seasonal, and residual components (e.g., STL).
* **Differencing**: Removes seasonal patterns by subtracting values from the same season in the previous year.
* **Fourier Transformations**: Captures periodic fluctuations for complex seasonal patterns.
* **Dummy Variables**: Incorporates seasonality in regression models by creating seasonal indicators.
* **Seasonal Models**: Uses models like SARIMA and Holt-Winters to explicitly account for seasonality.

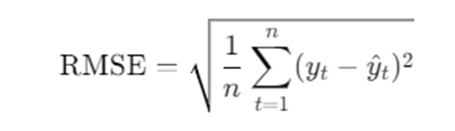
1. What are some common metrics for evaluating forecasting models (e.g., MAE, RMSE, MAPE)?

There are several key metrics used to evaluate the performance of forecasting models:

* **Mean Absolute Error (MAE):** This measures the average magnitude of the errors in a set of forecasts, without considering their direction. It is calculated as:



* **Root Mean Squared Error (RMSE):** This measures the square root of the average of squared differences between predicted and actual values. It gives a higher weight to larger errors



* **Mean Absolute Percentage Error (MAPE):** This metric expresses the error as a percentage of the actual values, making it easy to interpret:

