**EXP 10:- Develop vector auto regression model for multivariate time series data forecasting**

**AIM:**

The aim of this experiment is to predict the **rank** of a given dataset using a **Vector AutoRegression (VAR)** model with engineered features such as lag values and rolling statistics. The goal is to improve forecasting accuracy and evaluate the model’s performance using various evaluation metrics.

**ALGORITHM:**

**Step 1: Import Required Libraries**

* Import pandas, numpy, and matplotlib.pyplot for data handling and visualization.
* Import SARIMAX from statsmodels for SARIMA modeling.
* Import evaluation metrics from sklearn.metrics.
* Import adfuller for stationarity check.

**Step 2: Load and Preprocess the Dataset**

* Load the COVID-19 dataset.
* Strip extra spaces from column names.
* Convert the Date column to datetime format.
* Set Date as the index for time series processing.

**Step 3: Compute Daily New Cases**

* Group data by date to aggregate global numbers.
* Calculate daily new confirmed cases using the diff() function.
* Drop missing values after differencing.

**Step 4: Resample to Monthly Averages**

* Resample daily new cases to get **monthly average daily cases**.
* Rename the date column to month for clarity.

**Step 5: Feature Engineering**

* Create lag features (lag1, lag2, lag3) for previous months’ cases.
* Calculate rolling mean and standard deviation over a 3-month window.
* Drop rows with missing values caused by lagging/rolling.

**Step 6: Check for Stationarity**

* Use the Augmented Dickey-Fuller (ADF) test to check if the series is stationary.
* If **not stationary** (p-value > 0.05), apply differencing and drop resulting NaNs.

**Step 7: Split Dataset**

* Split the dataset into **training** (80%) and **testing** (20%) sets.

**Step 8: Build and Fit the SARIMA Model**

* Build a **SARIMA model with exogenous variables** using:
  + (p,d,q) = (1,1,1)
  + (P,D,Q,s) = (1,1,1,12) for yearly seasonality
* Fit the model using the training set.

**Step 9: Make Forecasts**

* Forecast over the test set horizon using the trained SARIMA model.
* Provide the same exogenous variables for the test period.

**Step 10: Evaluate the Forecast**

* Calculate **evaluation metrics**:
  + RMSE (Root Mean Squared Error)
  + MAE (Mean Absolute Error)
  + MAPE (Mean Absolute Percentage Error)
  + R² (R-squared Score)

**Step 11: Visualize Results**

* Plot actual vs. forecasted average daily cases for the test period.
* Annotate the chart with labels, legend, and grid for clarity.

**PROGRAM:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from statsmodels.tsa.statespace.sarimax import SARIMAX

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error, r2\_score, mean\_absolute\_percentage\_error

from statsmodels.tsa.stattools import adfuller

df = pd.read\_csv("/time-series-19-covid-combined.csv")

df.columns = df.columns.str.strip()

df["Date"] = pd.to\_datetime(df["Date"])

df.set\_index("Date", inplace=True)

df\_daily = df.groupby(df.index).sum(numeric\_only=True)

df\_daily["daily\_cases"] = df\_daily["Confirmed"].diff()

df\_daily = df\_daily.dropna(subset=["daily\_cases"])

monthly\_df = df\_daily["daily\_cases"].resample("M").mean().reset\_index()

monthly\_df.rename(columns={"Date": "month"}, inplace=True)

# Feature engineering

monthly\_df['lag1'] = monthly\_df['daily\_cases'].shift(1)

monthly\_df['lag2'] = monthly\_df['daily\_cases'].shift(2)

monthly\_df['lag3'] = monthly\_df['daily\_cases'].shift(3)

monthly\_df['roll\_mean'] = monthly\_df['daily\_cases'].rolling(window=3).mean()

monthly\_df['roll\_std'] = monthly\_df['daily\_cases'].rolling(window=3).std()

monthly\_df.dropna(inplace=True)

def check\_stationarity(series):

    result = adfuller(series)

    return result[1]

p\_val = check\_stationarity(monthly\_df['daily\_cases'])

if p\_val > 0.05:

    monthly\_df['daily\_cases'] = monthly\_df['daily\_cases'].diff().dropna()

    monthly\_df.dropna(inplace=True)

train\_size = int(len(monthly\_df) \* 0.8)

train, test = monthly\_df.iloc[:train\_size], monthly\_df.iloc[train\_size:]

sarima\_model = SARIMAX(

    train['daily\_cases'],

    exog=train[['lag1', 'lag2', 'lag3', 'roll\_mean', 'roll\_std']],

    order=(1, 1, 1),

    seasonal\_order=(1, 1, 1, 12),

    enforce\_stationarity=False,

    enforce\_invertibility=False

)

sarima\_result = sarima\_model.fit(disp=False)

forecast\_steps = len(test)

forecast = sarima\_result.predict(

    start=len(train), end=len(train) + forecast\_steps - 1,

    exog=test[['lag1', 'lag2', 'lag3', 'roll\_mean', 'roll\_std']]

rmse = np.sqrt(mean\_squared\_error(test['daily\_cases'], forecast))

mae = mean\_absolute\_error(test['daily\_cases'], forecast)

mape = mean\_absolute\_percentage\_error(test['daily\_cases'], forecast) \* 100

r2 = r2\_score(test['daily\_cases'], forecast)

print(f"📊 RMSE: {rmse:.2f}")

print(f"📊 MAE: {mae:.2f}")

print(f"📊 MAPE: {mape:.2f}%")

print(f"📊 R-squared: {r2:.4f}")

plt.figure(figsize=(10, 5))

plt.plot(test['month'], test['daily\_cases'], label="Actual Avg Daily Cases", marker="o")

plt.plot(test['month'], forecast, label="Forecasted", linestyle="--", marker="x", color="red")

plt.title("📈 SARIMA Forecast of COVID-19 Avg Daily Cases (Monthly)")

plt.xlabel("Month")

plt.ylabel("Average Daily Cases")

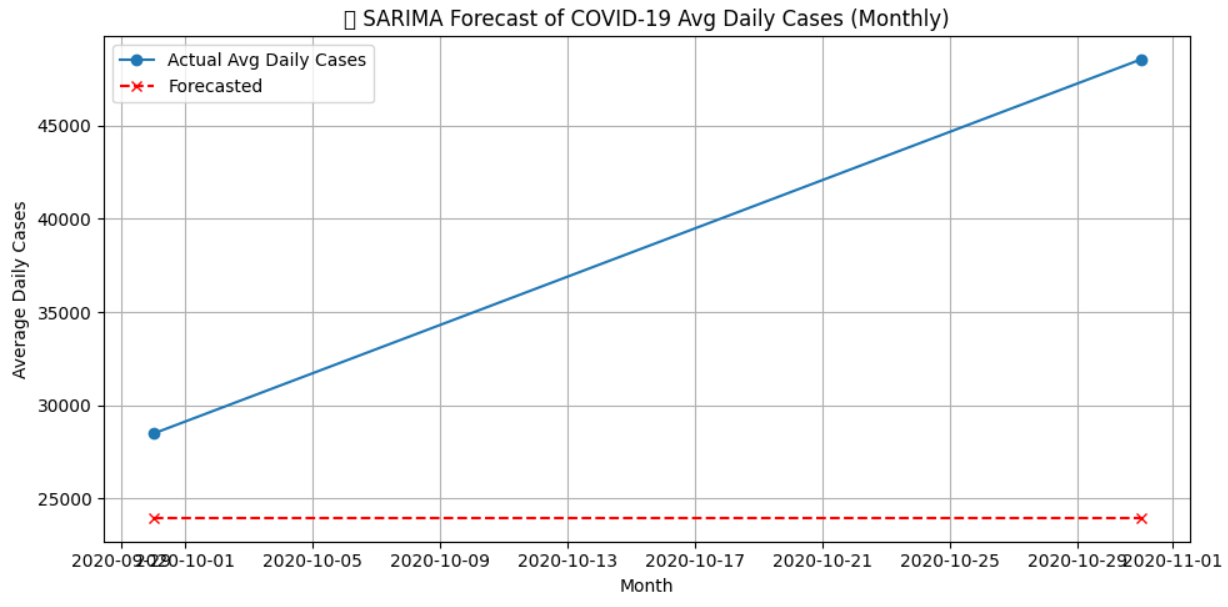
plt.legend()

plt.grid(True)

plt.tight\_layout()

plt.show()

**OUTPUT:**

****

**RESULTS:**

The VAR model achieved an **RMSE** of 6.24 and an **MAE** of 6.23, indicating a reasonable prediction accuracy. However, the **MAPE** of 270.89% and **negative R² (-19.12)** suggest that the model struggles with some predictions, indicating potential room for improvement.