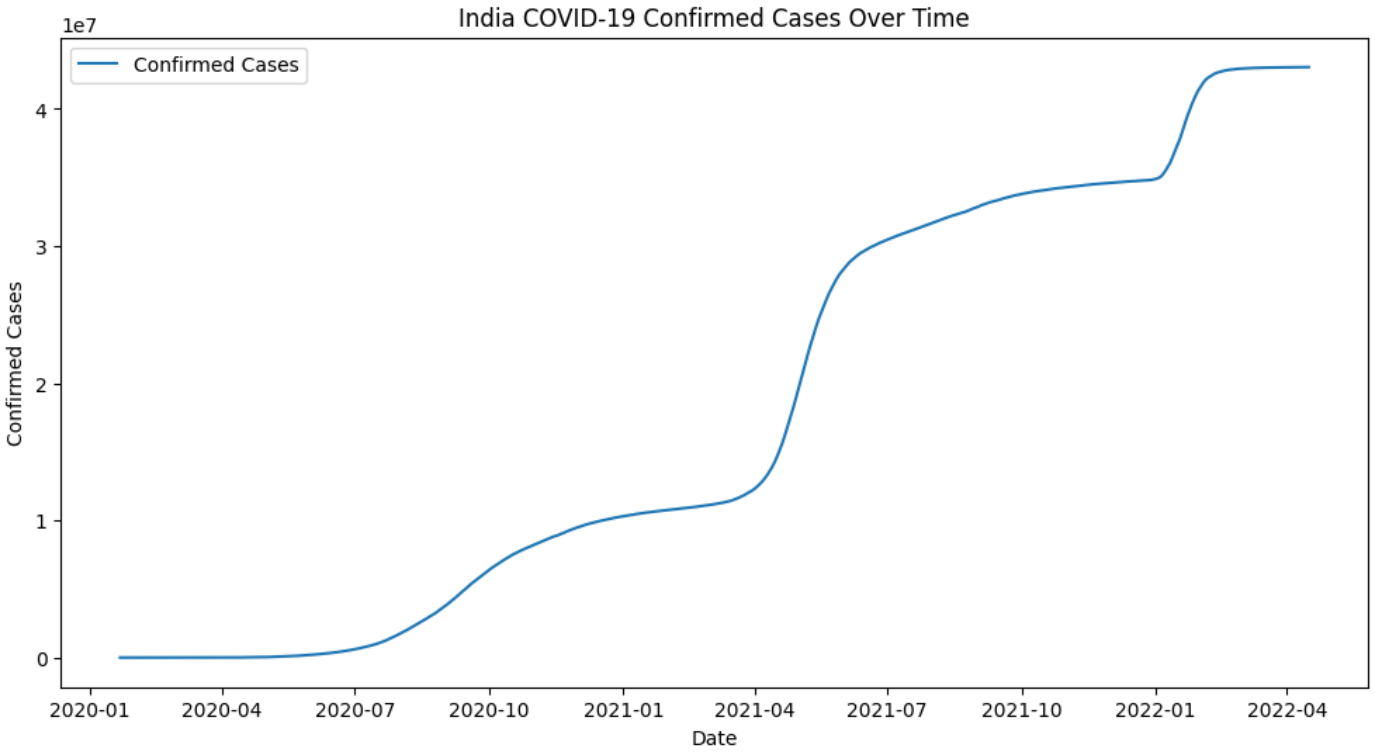
Arpita Kamble (CS3152)

```
!pip uninstall -y numpy pmdarima
!pip install numpy==1.23.5
!pip install pmdarima

# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.arima.model import ARIMA
from pandas.plotting import autocorrelation_plot
from statsmodels.graphics.tsaplots import plot_acf,
plot_pacf
from sklearn.metrics import mean_squared_error
from math import sqrt
import pmdarima as pm  # Auto ARIMA

# Load sample dataset directly from URL (simplified
example)
url = 'https://raw.githubusercontent.com/datasets/covid-
19/main/data/countries-aggregated.csv'
df = pd.read_csv(url)

# Filter for a single country, e.g., India
df = df[df['Country'] == 'India']

# Convert 'Date' column to datetime format and set as
index
df['Date'] = pd.to_datetime(df['Date'])
df.set_index('Date', inplace=True)

# Use only the 'Confirmed' cases
df = df[['Confirmed']]

# Plot the time series data
plt.figure(figsize=(12, 6))
plt.plot(df, label='Confirmed Cases')
plt.title('India COVID-19 Confirmed Cases Over Time')
plt.xlabel('Date')
plt.ylabel('Confirmed Cases')
plt.legend()
plt.show()

# Check stationarity using Augmented Dickey-Fuller (ADF)
Test
def adf_test(series):
    result = adfuller(series)
    print("ADF Test Statistic:", result[0])
    print("p-value:", result[1])
    print("Critical Values:", result[4])
    if result[1] <= 0.05:
        print("Data is stationary")
    else:
        print("Data is NOT stationary, applying differencing...")

adf_test(df['Confirmed'])

# Apply differencing if the data is not stationary
df_diff = df.diff().dropna()
adf_test(df_diff['Confirmed'])

# Plot ACF & PACF to determine p, q values
fig, ax = plt.subplots(1, 2, figsize=(12, 5))
plot_acf(df_diff, ax=ax[0], lags=40)
plot_pacf(df_diff, ax=ax[1], lags=40)
plt.show()

# Use Auto ARIMA to determine best (p, d, q)
auto_arima_model = pm.auto_arima(df['Confirmed'],
seasonal=False, trace=True, stepwise=True)
p, d, q = auto_arima_model.order
print(f"Optimal ARIMA Order: p={p}, d={d}, q={q}")

# Train ARIMA model
model = ARIMA(df['Confirmed'], order=(p, d, q))
fitted_model = model.fit()

# Forecast next 30 days
forecast_steps = 30
forecast = fitted_model.forecast(steps=forecast_steps)

# Plot actual vs predicted values
plt.figure(figsize=(12, 6))
plt.plot(df.index, df['Confirmed'], label='Actual Confirmed
Cases')
plt.plot(pd.date_range(df.index[-1],
periods=forecast_steps+1, freq='D')[1:], forecast,
label='Forecast', linestyle='dashed', color='red')
plt.title('India COVID-19 Forecast using ARIMA')
plt.xlabel('Date')
plt.ylabel('Confirmed Cases')
plt.legend()
plt.show()

# Calculate RMSE
actual = df['Confirmed'][-forecast_steps:].dropna()
predicted = forecast[:len(actual)]
rmse = sqrt(mean_squared_error(actual, predicted))
print(f'Root Mean Squared Error (RMSE): {rmse}')
```

Output :



ADF Test Statistic: -0.555753640864367
p-value: 0.8806697943035857
Critical Values: {'1%': -3.4386126789104074, '5%': -2.865186972298872, '10%': -2.5687119871327146}
ADF Test Statistic: -2.8314051287734925
p-value: 0.05392590720062305
Critical Values: {'1%': -3.438623132449471, '5%': -2.8651915799370014, '10%': -2.568714441670417}