TECHNICAL REPORT: COVID-19 DATA ANALYSIS

Title: COVID-19 Data Analysis: Trends, Insights, and Implications

Author: OPARAUGO IHEOMA A

Date: 29/11/2024

INTRODUCTION

The COVID-19 pandemic significantly impacted global health, economics, and policy. This report analyzes three datasets: day_wise.csv, country_wise.csv, and covid_19_clean.csv, providing insights into global trends, geographic disparities, and predictive modeling of case progression. The report covers data preparation, exploratory data analysis (EDA), model development, model evaluation, and key insights.

DATA PREPARATION

Overview of Datasets

- **day_wise.csv**: Tracks daily global statistics.
    - Key columns: Date, Confirmed, Deaths, Recovered, Active.
- **country_wise.csv**: Contains aggregated country-level data.
    - Key columns: Country/Region, Confirmed, Deaths, Recovered, Active, Mortality Rate.
- **covid_19_clean.csv**: Offers detailed records by country and province.
    - Key columns: Country/Region, Province/State, Date, Confirmed, Deaths, Recovered.

Data Cleaning Steps

- Checked for missing values and imputed/reviewed where necessary.
- Ensured consistent formatting of dates across all datasets.
- Aggregated regional data in covid_19_clean.csv for country-level comparisons.
- Created derived features:
    - **Mortality Rate** = (Deaths / Confirmed) × 100 ○ **Recovery Rate** = (Recovered / Confirmed) × 100 etc

Data Cleaning Summary

- Missing values in **Province/State** (34,404) were replaced with `"Unknown"`.
- The **Date** column was successfully standardized to datetime format.
- No duplicates were found or removed.

[31]:
```
# Feature Engineering

# Sort data by Country/Region and Date for consistency
data = data.sort_values(by=["Country/Region", "Date"])

# Calculate daily growth rates for Confirmed cases
data["Daily Growth Rate"] = data.groupby("Country/Region")["Confirmed"].diff().fillna(0)

# Calculate mortality rate (Deaths / Confirmed) * 100
data["Mortality Rate"] = (data["Deaths"] / data["Confirmed"]).replace([float("inf"), -float("inf")], 0).fillna(0) * 100

# Assume a hypothetical population for cases per population analysis (if not given, default to 1M per country)
# Since population data isn't included, we'll use a placeholder value for demonstration
population_placeholder = 1_000_000
data["Cases Per Population"] = data["Confirmed"] / population_placeholder

# Preview the dataset after feature engineering
data[["Date", "Country/Region", "Confirmed", "Daily Growth Rate", "Mortality Rate", "Cases Per Population", "Continent"]].head()
```

[31]:

	Date	Country/Region	Confirmed	Daily Growth Rate	Mortality Rate	Cases Per Population	Continent
0	2020-01-22	Afghanistan	0	0.0	0.0	0.0	Eastern Mediterranean
261	2020-01-23	Afghanistan	0	0.0	0.0	0.0	Eastern Mediterranean
522	2020-01-24	Afghanistan	0	0.0	0.0	0.0	Eastern Mediterranean
783	2020-01-25	Afghanistan	0	0.0	0.0	0.0	Eastern Mediterranean
1044	2020-01-26	Afghanistan	0	0.0	0.0	0.0	Eastern Mediterranean

Tools Used

- Python: Pandas, Numpy, Matplotlib, Seaborn, Scikit-learn.

Exploratory Data Analysis (EDA)

Global Trends (day_wise.csv)

- **Objective**: Understand the progression of COVID-19 globally over time.
- **Insights**:
  - Cases (Confirmed, Active, Recovered, Deaths) show exponential growth starting midMarch 2020.
  - Active cases peaked earlier, with deaths rising after a delay.

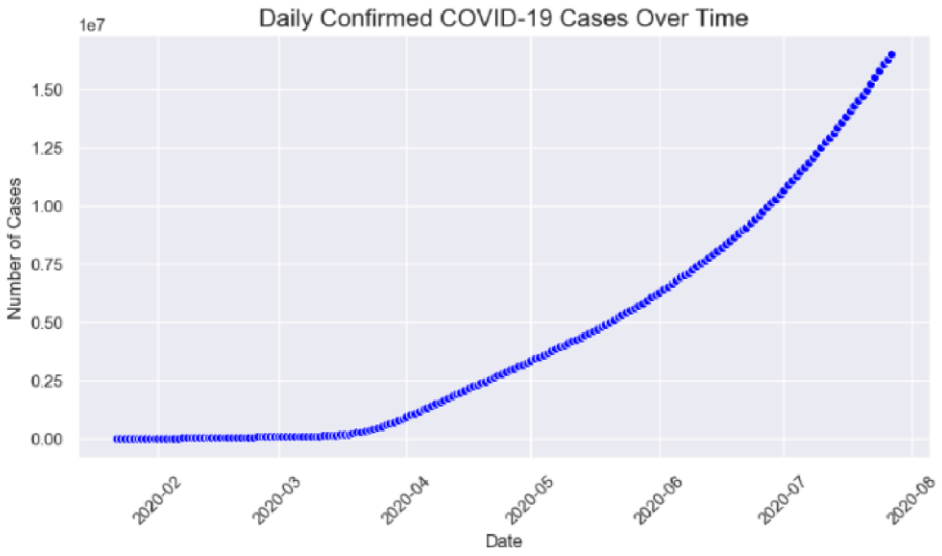
Visualization:

- Line plot showing trends of Confirmed, Active, Recovered, and Death cases over time

4. Univariant Analysis

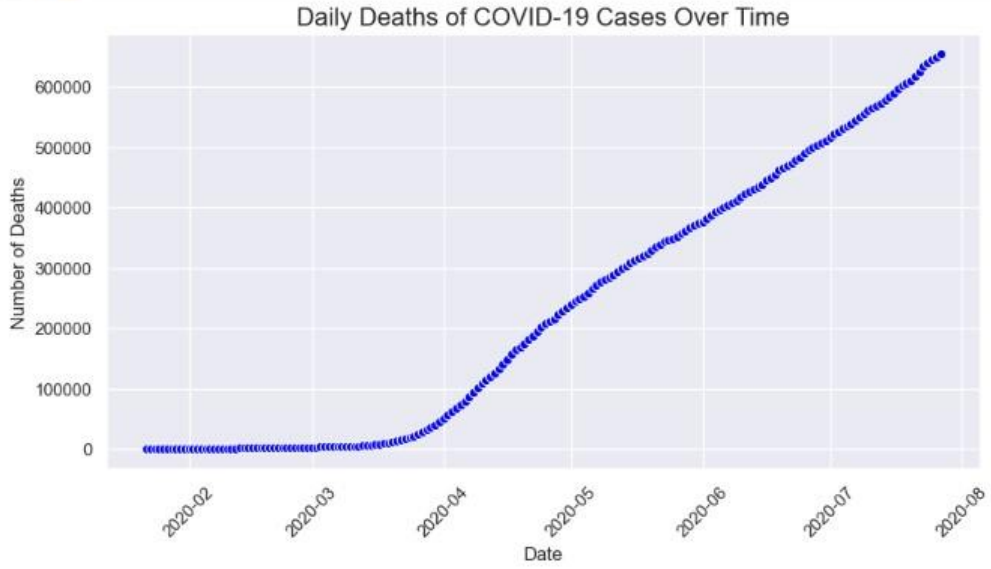
```
3]: # Daily Trends: Confirmed Cases Over Time
daily_trends = data.groupby("Date")["Confirmed"].sum()

plt.figure(figsize=(10, 5))
sns.set_theme(style="darkgrid")
sns.lineplot(data=daily_trends, marker="o", color="blue")
plt.title("Daily Confirmed COVID-19 Cases Over Time", fontsize=16)
plt.xlabel("Date", fontsize=12)
plt.ylabel("Number of Cases", fontsize=12)
plt.xticks(rotation=45)
plt.show()
```



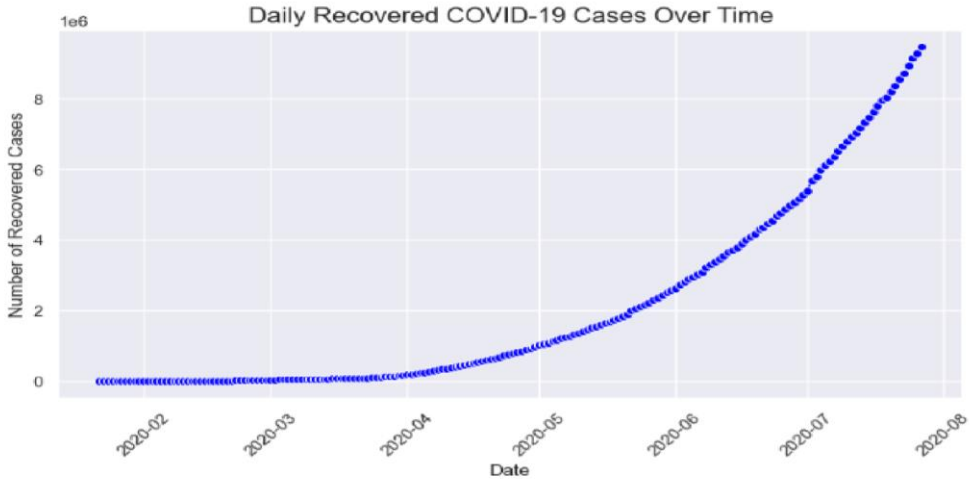

```
14]: # Daily Trends: Confirmed Cases Over Time
daily_trends = data.groupby("Date")["Deaths"].sum()

plt.figure(figsize=(10, 5))
sns.set_theme(style="darkgrid")
sns.lineplot(data=daily_trends, marker="o", color="blue")
plt.title("Daily Deaths of COVID-19 Cases Over Time", fontsize=16)
plt.xlabel("Date", fontsize=12)
plt.ylabel("Number of Deaths", fontsize=12)
plt.xticks(rotation=45)
plt.show()
```



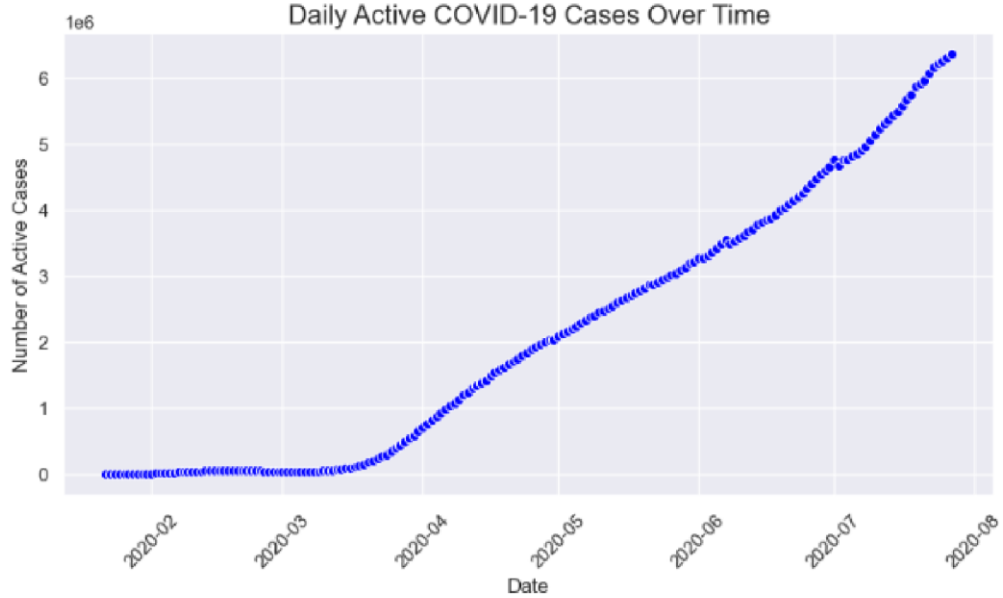
[15]:
```
# Daily Trends: Confirmed Cases Over Time
daily_trends = data.groupby("Date")["Recovered"].sum()

plt.figure(figsize=(10, 5))
sns.set_theme(style="darkgrid")
sns.lineplot(data=daily_trends, marker="o", color="blue")
plt.title("Daily Recovered COVID-19 Cases Over Time", fontsize=16)
plt.xlabel("Date", fontsize=12)
plt.ylabel("Number of Recovered Cases", fontsize=12)
plt.xticks(rotation=45)
plt.show()
```



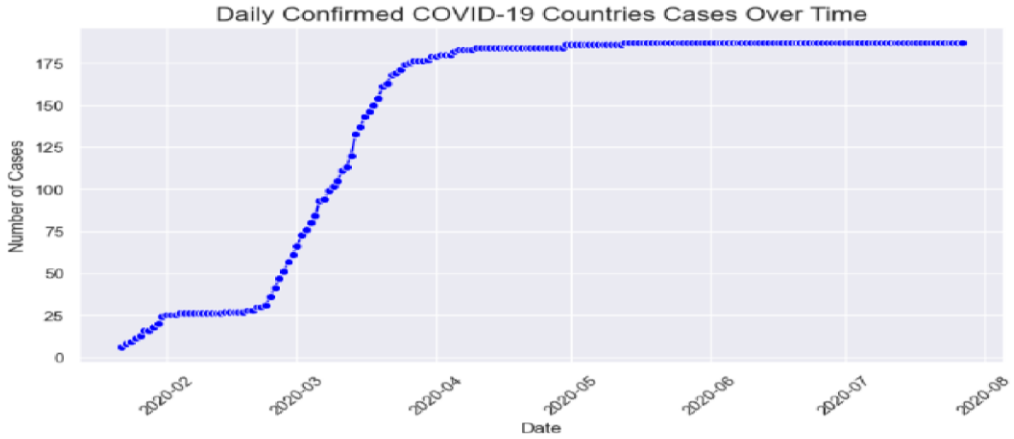
[16]:
```
# Daily Trends: Confirmed Cases Over Time
daily_trends = data.groupby("Date")["Active"].sum()

plt.figure(figsize=(10, 5))
sns.set_theme(style="darkgrid")
sns.lineplot(data=daily_trends, marker="o", color="blue")
plt.title("Daily Active COVID-19 Cases Over Time", fontsize=16)
plt.xlabel("Date", fontsize=12)
plt.ylabel("Number of Active Cases", fontsize=12)
plt.xticks(rotation=45)
plt.show()
```

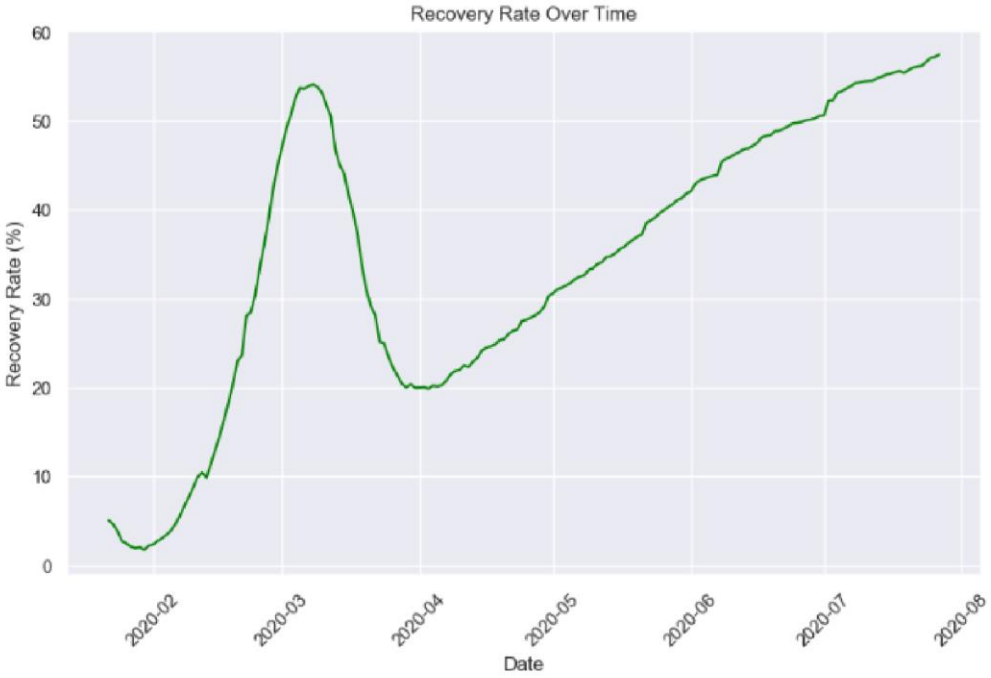


[17]:
```
# Daily Trends: Number of Countries Affected Per Day
daily_trends = data.groupby("Date")["No. of countries"].sum()

plt.figure(figsize=(10, 5))
sns.set_theme(style="darkgrid")
sns.lineplot(data=daily_trends, marker="o", color="blue")
plt.title("Daily Confirmed COVID-19 Countries Cases Over Time", fontsize=16)
plt.xlabel("Date", fontsize=12)
plt.ylabel("Number of Cases", fontsize=12)
plt.xticks(rotation=45)
plt.show()
```



[21]:
```
# Calculate Recovery Rate
data['Recovery_Rate'] = (data['Recovered'] / data['Confirmed']) * 100
plt.figure(figsize=(10, 6))
sns.lineplot(x=data.index, y=data['Recovery_Rate'], color='green')
plt.title('Recovery Rate Over Time')
plt.xlabel('Date')
plt.ylabel('Recovery Rate (%)')
plt.xticks(rotation=45)
plt.show()
```



Exploratory Data Analysis (EDA)

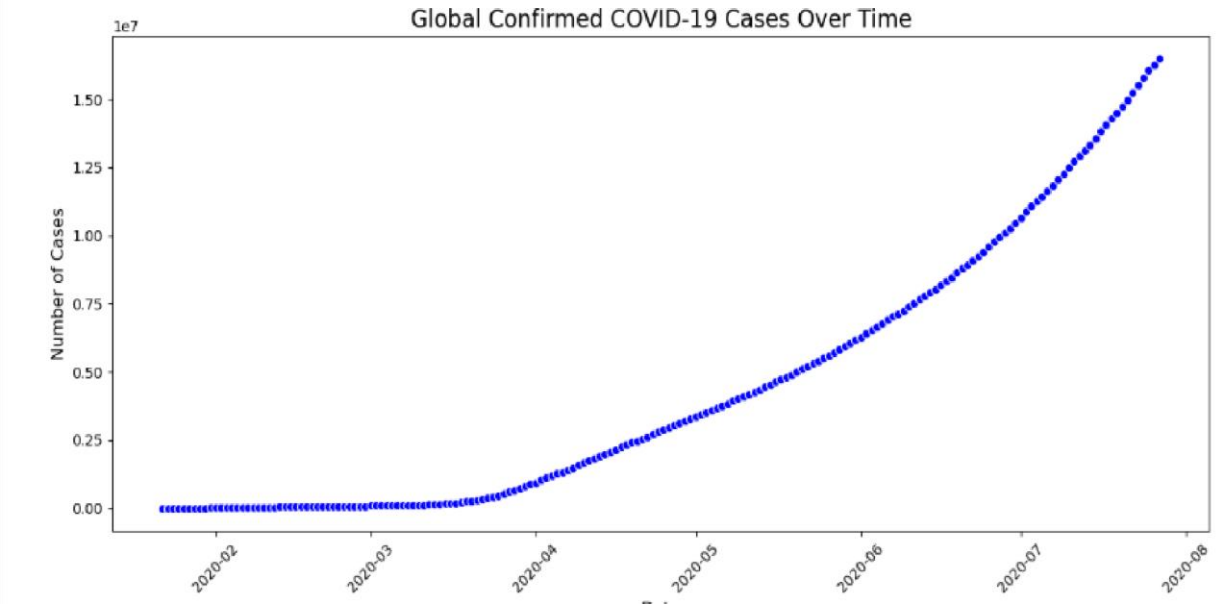
1. Global Trends of Confirmed Cases Over Time ¶

Here's how you can perform the visualization on your local system:

Use the following code to visualize the trend:

[32]:
```
# Global Trends: Confirmed Cases Over Time
global_trends = data.groupby("Date")["Confirmed"].sum()

plt.figure(figsize=(12, 6))
sns.lineplot(data=global_trends, marker="o", color="blue")
plt.title("Global Confirmed COVID-19 Cases Over Time", fontsize=16)
plt.xlabel("Date", fontsize=12)
plt.ylabel("Number of Cases", fontsize=12)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Country-Level Analysis (country_wise.csv)

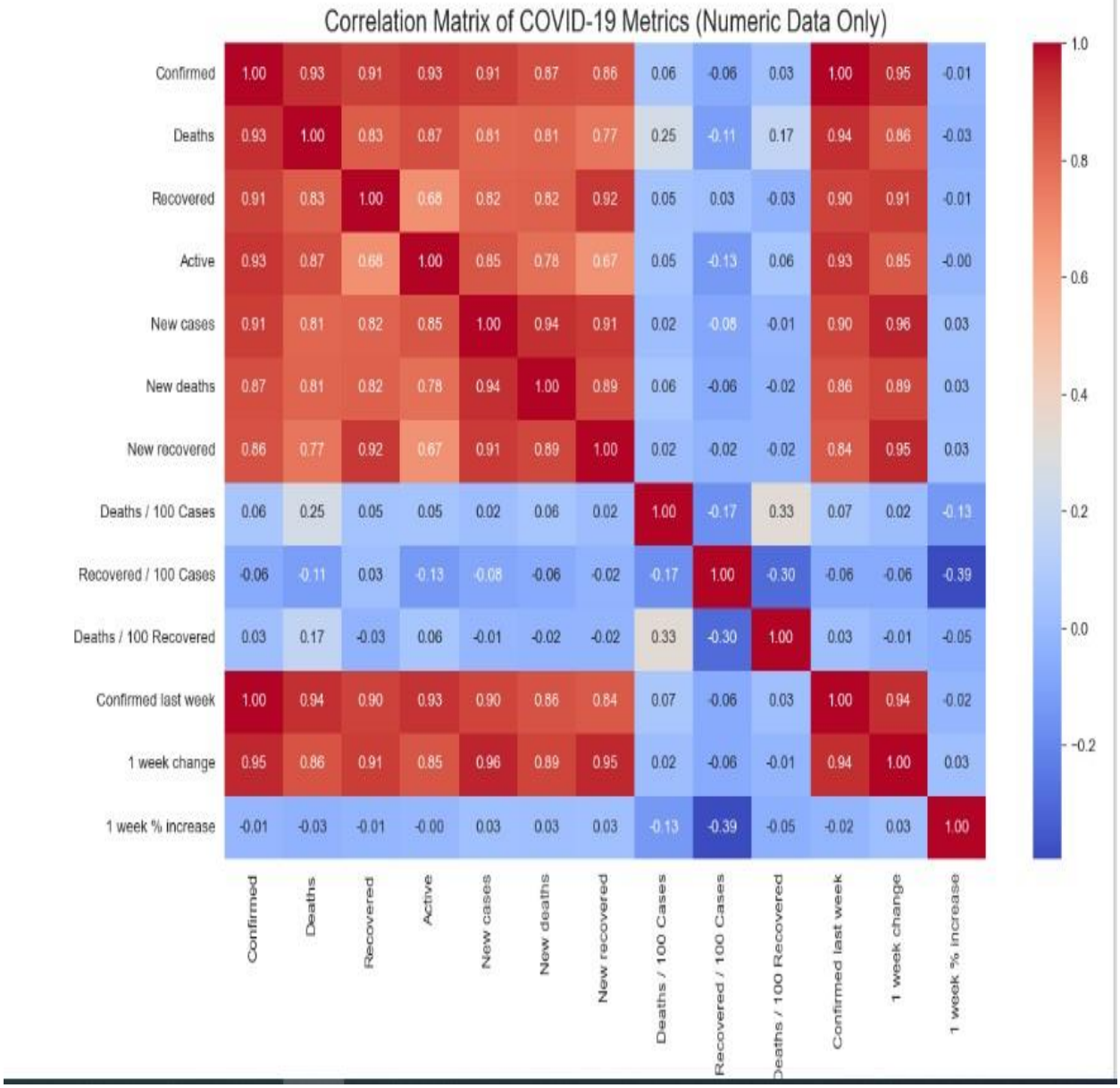
- **Objective**: Examine disparities in COVID-19 impact across countries. ▪ **Insights**:
    - The US, India, and Brazil reported the highest confirmed cases globally.
    - Mortality rates varied significantly, with Europe and Latin America experiencing higher rates compared to Asia.

Visualization:

- Bar chart of the top 10 countries by Confirmed cases.
- Heatmap for Mortality and Recovery rates across countries.

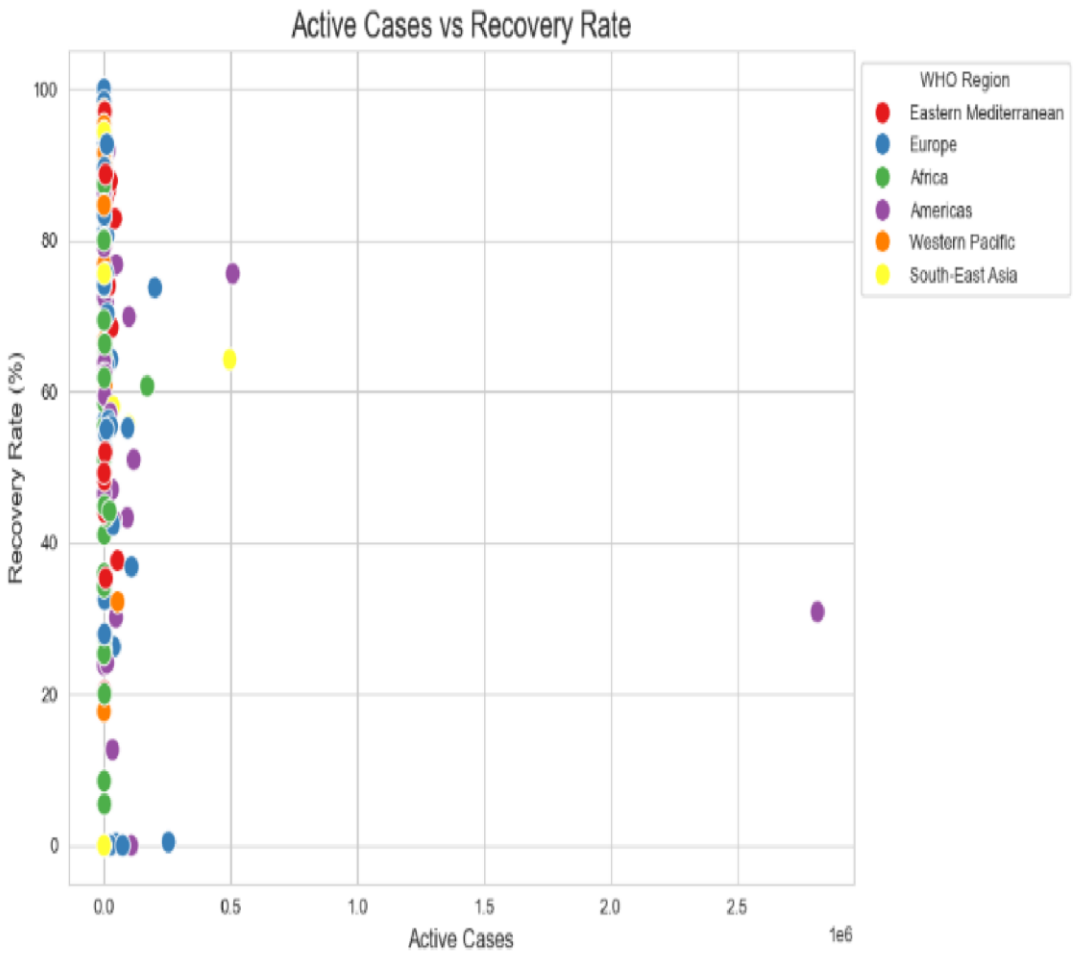
[61]:

```
# Replot the heatmap with fixed data
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix_fixed, annot=True, fmt=".2f", cmap="coolwarm", cbar=True)
plt.title("Correlation Matrix of COVID-19 Metrics (Numeric Data Only)", fontsize=16)
plt.tight_layout()
plt.show()
```



[57]:

```
# Scatter plot: Active cases vs Recovery rate
plt.figure(figsize=(10, 6))
sns.scatterplot(x="Active", y="Recovered / 100 Cases", data=data, hue="WHO Region", palette="Set1", s=100)
plt.title("Active Cases vs Recovery Rate", fontsize=16)
plt.xlabel("Active Cases", fontsize=12)
plt.ylabel("Recovery Rate (%)", fontsize=12)
plt.legend(title="WHO Region", loc="upper left", bbox_to_anchor=(1, 1))
plt.tight_layout()
plt.show()
```

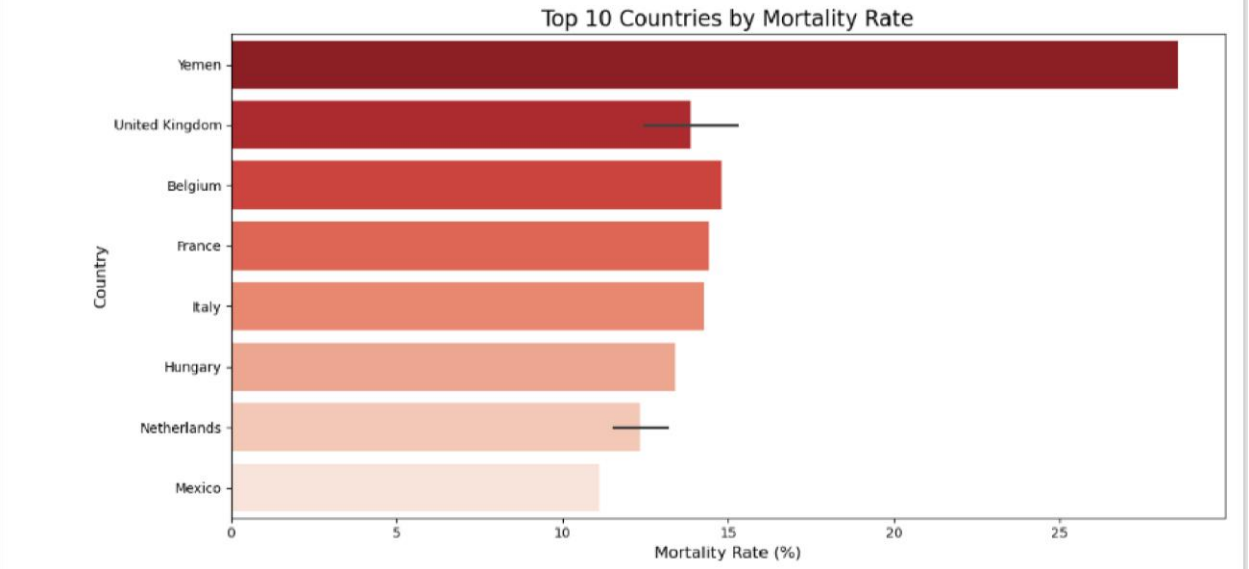


2. Mortality Rate Across Countries

Visualize the top 10 countries with the highest mortality rates:

[33]:
```
# Top 10 Countries by Mortality Rate
latest_data = data[data["Date"] == data["Date"].max()]
top_countries = latest_data.nlargest(10, "Mortality Rate")[["Country/Region", "Mortality Rate"]]

plt.figure(figsize=(12, 6))
sns.barplot(data=top_countries, x="Mortality Rate", y="Country/Region", palette="Reds_r")
plt.title("Top 10 Countries by Mortality Rate", fontsize=16)
plt.xlabel("Mortality Rate (%)", fontsize=12)
plt.ylabel("Country", fontsize=12)
plt.tight_layout()
plt.show()
```



Detailed Regional Analysis (covid_19_clean.csv)

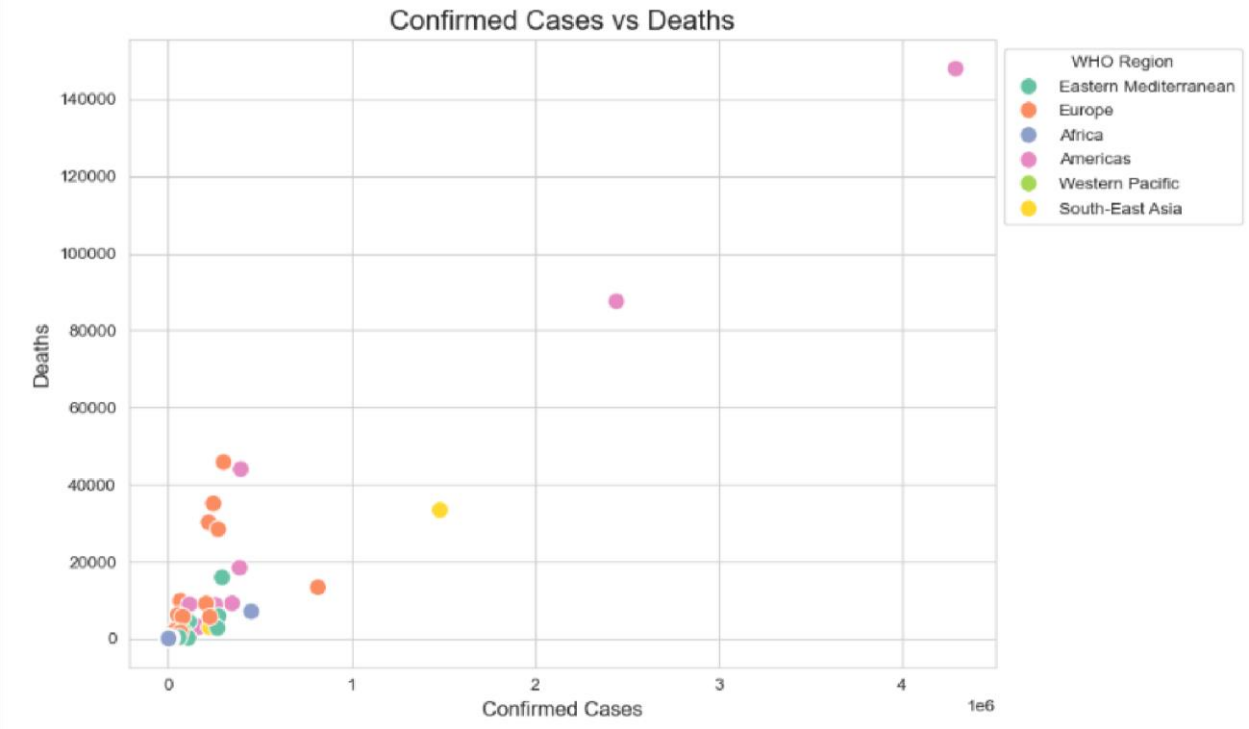
- **Objective**: Analyze province-level data for high-impact regions. ▪ **Insights**:
    - Provinces in countries like the US and India exhibited significant disparities in case distribution.
    - Urban areas consistently reported higher confirmed cases and mortality rates.

Visualization:

- Scatter plot of Confirmed cases vs. Deaths by region.

```
correlation_matrix_fixed

# Scatter plot: Confirmed vs Deaths
plt.figure(figsize=(10, 6))
sns.scatterplot(x="Confirmed", y="Deaths", data=data, hue="WHO Region", palette="Set2", s=100)
plt.title("Confirmed Cases vs Deaths", fontsize=16)
plt.xlabel("Confirmed Cases", fontsize=12)
plt.ylabel("Deaths", fontsize=12)
plt.legend(title="WHO Region", loc="upper left", bbox_to_anchor=(1, 1))
plt.tight_layout()
plt.show()
```



Model Development

Objective

- Develop predictive models to forecast number of cases over time using historical data.
- To predict weather a country is at risk or not using historical data

Features and Target Variables

- **Features**: Mortality Rate, Recovery Rate, region-specific indicators, Cases Per Population etc
- **Target**: Binary label (e.g., high-risk if confirmed cases > threshold

Model Selection

- **Random Forest Classifier**

Training and Validation

- Data split into training (70%) and testing (30%) sets. Cross-validation (3-fold) to assess generalization.

Model Evaluation

Metrics

- **Root-Mean Squared Error (RMSE)**: Squared Average error magnitude.
- Use metrics like Accuracy, Precision, Recall, and F1-Score.

Results

- **Random Forest** did exceptionally well.

Interpreting Classification Results

- **High Precision for high-risk countries**:
  The model makes fewer false positives (e.g., doesn't wrongly classify low-risk countries as high-risk).

- **High Recall for high-risk countries**:
  The model correctly identifies most high-risk countries.

- **Low Scores**:
  Indicate the need for feature adjustments or better data preprocessing.

Classification Report

The output of classification report:

- **Precision**: How many predicted high-risk countries were actually high-risk.
- **Recall**: How many actual high-risk countries were correctly identified.
- **F1-Score**: Balance of precision and recall (1.0 is perfect).
- **Support**: Number of samples in each class.

1. Time-Series Model Development

We'll predict the number of confirmed cases over time using models like ARIMA.

Steps to Develop a Time-Series Model:

1. Prepare the Data: ¶

- Focus on the Date and Confirmed columns.
- Aggregate the data globally or per country, depending on the prediction scope.

2. Split the Data:

- Use 80% of the data for training and 20% for testing.

3. Model Training:

- Use ARIMA for basic predictions.

4. Evaluate the Model:

- Use metrics like **Mean Absolute Error (MAE)** or **Root Mean Squared Error (RMSE)**.

[42]:
```
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_squared_error

# Aggregate data by date
global_data = data.groupby("Date")["Confirmed"].sum()

# Split into training and testing sets
train_size = int(len(global_data) * 0.8)
train, test = global_data[:train_size], global_data[train_size:]

# Train ARIMA model
model = ARIMA(train, order=(5, 1, 0))
model_fit = model.fit()

# Make predictions
predictions = model_fit.forecast(steps=len(test))
rmse = mean_squared_error(test, predictions, squared=False)

print(f"RMSE: {rmse}")
```

```
RMSE: 773632.9709797429
```

Classification Report

The output of `classification_report` will look like this:

- **Precision**: How many predicted high-risk countries were actually high-risk.
- **Recall**: How many actual high-risk countries were correctly identified.
- **F1-Score**: Balance of precision and recall (1.0 is perfect).
- **Support**: Number of samples in each class.

[44]:
```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report

# Create a binary target variable
latest_data = data[data["Date"] == data["Date"].max()]
latest_data["High-Risk"] = (latest_data["Confirmed"] > 100000).astype(int)

# Define features and target
X = latest_data[["Mortality Rate", "Cases Per Population"]]
y = latest_data["High-Risk"]

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Train Random Forest Classifier
clf = RandomForestClassifier(random_state=42)
clf.fit(X_train, y_train)

# Evaluate the model
y_pred = clf.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.97      1.00      0.99        70
           1       1.00      0.78      0.88         9

    accuracy                           0.97        79
   macro avg       0.99      0.89      0.93        79
weighted avg       0.98      0.97      0.97        79
```

Visualization:

- Line plot of the model.

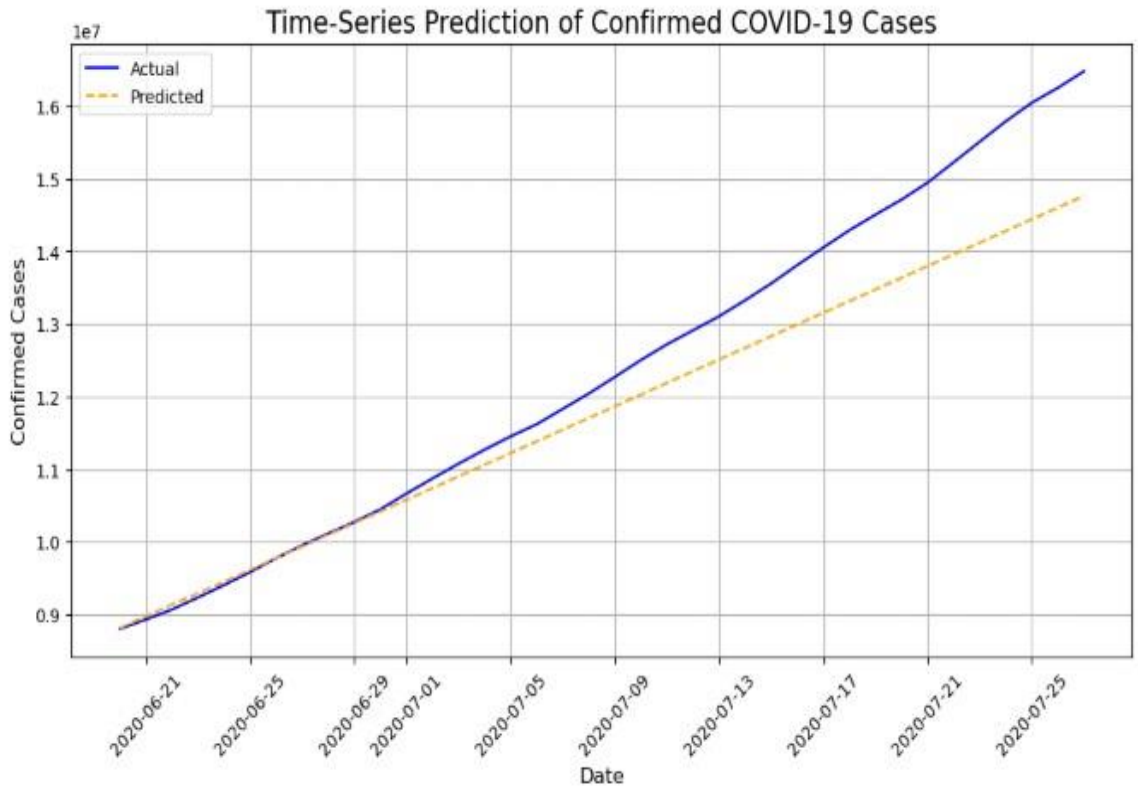
1. Visualizing and Interpreting the Time-Series Model

Visualization of Predictions vs. Actuals

This will help you compare the model's forecast against the actual number of cases.

```
3]: import matplotlib.pyplot as plt

# Plot Actual vs. Predicted
plt.figure(figsize=(12, 6))
plt.plot(test.index, test, label="Actual", color="blue")
plt.plot(test.index, predictions, label="Predicted", color="orange", linestyle="--")
plt.title("Time-Series Prediction of Confirmed COVID-19 Cases", fontsize=16)
plt.xlabel("Date", fontsize=12)
plt.xticks(rotation=45)
plt.ylabel("Confirmed Cases", fontsize=12)
plt.legend()
plt.grid(True)
plt.show()
```



2. Visualizing and Interpreting the Classification Model

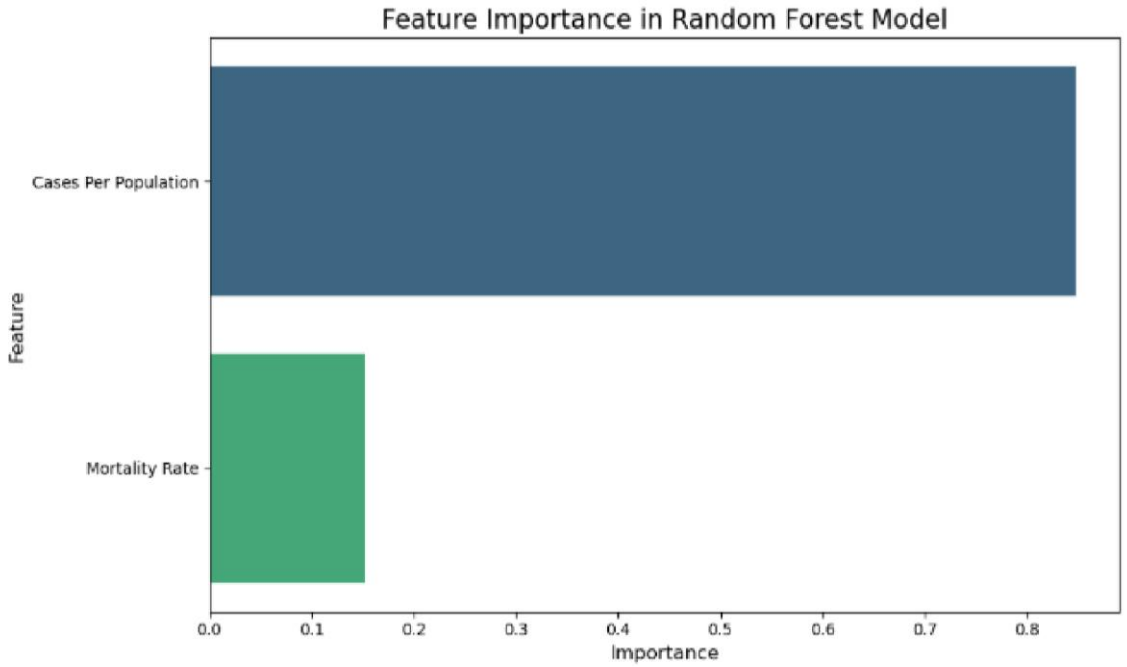
Visualization: Feature Importance

Random Forest can tell us which features were most important in making predictions.

```
[5]: import seaborn as sns
import pandas as pd

# Feature Importance Plot
feature_importances = pd.DataFrame({
    'Feature': X.columns,
    'Importance': clf.feature_importances_
}).sort_values(by='Importance', ascending=False)

plt.figure(figsize=(10, 6))
sns.barplot(data=feature_importances, x="Importance", y="Feature", palette="viridis")
plt.title("Feature Importance in Random Forest Model", fontsize=16)
plt.xlabel("Importance", fontsize=12)
plt.ylabel("Feature", fontsize=12)
plt.tight_layout()
plt.show()
```



Key Insights and Public Health Implications

Global and Regional Trends

- Exponential growth in early 2020 highlights the rapid global spread of the virus.
- Active case peaks are strong indicators of subsequent healthcare system strain.

Geographic Disparities

- Countries with better healthcare access showed lower mortality rates.
- Socioeconomic disparities contributed to regional variations in outcomes.

Policy Recommendations

- Prioritize vaccine distribution to regions with high Active and Death case correlations.
- Allocate healthcare resources to areas with delayed recovery and higher mortality rates.

Conclusion

This analysis provides insights into the global and regional impacts of COVID-19, emphasizing the importance of timely intervention and resource allocation. Predictive models suggest that reducing Active cases early can significantly mitigate mortality. Future work could focus on socioeconomic factors and real-time data for policy refinement.