Step 1: Load Libraries and Dataset

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.cluster import KMeans
from sklearn.metrics import mean_squared_error, silhouette_score
```

Step 2: Load the Dataset

```
data = pd.read_csv("covid19_data.csv")
print(data.head())
```

Step 3: Data Preprocessing

```
# Check for missing values
print(data.isnull().sum())

# Fill missing values with forward fill method
data.fillna(method='ffill', inplace=True)

# Feature Engineering: Add active cases column
data['Active_Cases'] = data['Confirmed'] - (data['Recovered'] + data['Deaths'])

# Normalize numerical features
scaler = MinMaxScaler()
```

```
data[['Confirmed', 'Recovered', 'Deaths', 'Active_Cases']] = scaler.fit_transform(
    data[['Confirmed', 'Recovered', 'Deaths', 'Active_Cases']])
```

Step 4: Split Data for Machine Learning

```
# Split into features (X) and target (y)
X = data[['Confirmed', 'Recovered', 'Deaths']]
y = data['Active_Cases']

# Split into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

Step 5: Linear Regression Model

```
# Train a linear regression model
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)

# Predictions and evaluation
y_pred = lr_model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print(f"Linear Regression MSE: {mse}")
```

Step 6: Random Forest Model

```
# Train a random forest regressor
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Predictions and evaluation
```

```
y_pred_rf = rf_model.predict(X_test)
mse_rf = mean_squared_error(y_test, y_pred_rf)
print(f"Random Forest MSE: {mse_rf}")
```

Step 7: K-Means Clustering (Optional)

```
# Prepare data for clustering
clustering_data = data[['Confirmed', 'Recovered', 'Deaths']]

# Apply KMeans clustering
kmeans = KMeans(n_clusters=3, random_state=42)
data['Cluster'] = kmeans.fit_predict(clustering_data)

# Evaluate clustering
silhouette = silhouette_score(clustering_data, data['Cluster'])
print(f"Silhouette Score: {silhouette}")
```