COVID-19 CASE ANALYSIS – PROJECT

Data Analytics with Cognos (DAC)

Phase 3 – Development Part I

Team Members:

Manoj. S – 2021115061

Mini Gnana Sekaran - 2021115062

Mohammed Nihal - 2021115063

Mridula. P - 2021115064

Dharsan. S - 2021115309

COVID-19 Case - Data Analysis Report

Executive Summary

This report provides an analysis of the daily death and recovery data for COVID-19 in Germany, France, and Italy. The analysis covers a specific time frame and is accompanied by visual representations.

TABLE OF CONTENTS

Introduction

The COVID-19 pandemic has had a significant impact on countries worldwide. In this report, we focus on Germany, France, and Italy, presenting an analysis of daily death and recovery data to gain insights into the progression of the pandemic.

Data Collection and Sources

The data used in this analysis was collected from reliable sources, such as government health agencies and international health organizations.

Data Preprocessing

Prior to analysis, the data underwent preprocessing, including cleaning, handling missing values, and transforming data formats. This ensured the accuracy and reliability of our analysis.

Data Analysis:

Daily Deaths:

We analyzed the daily death data to:

- Identify peak periods of fatalities.
- Assess the impact of government interventions.
- Determine trends in mortality rates.

Daily Recoveries:

We analyzed the daily recovery data to:

- Understand the rate of recovery.
- Identify patterns in recoveries.
- Assess the effectiveness of healthcare systems.

Key Findings

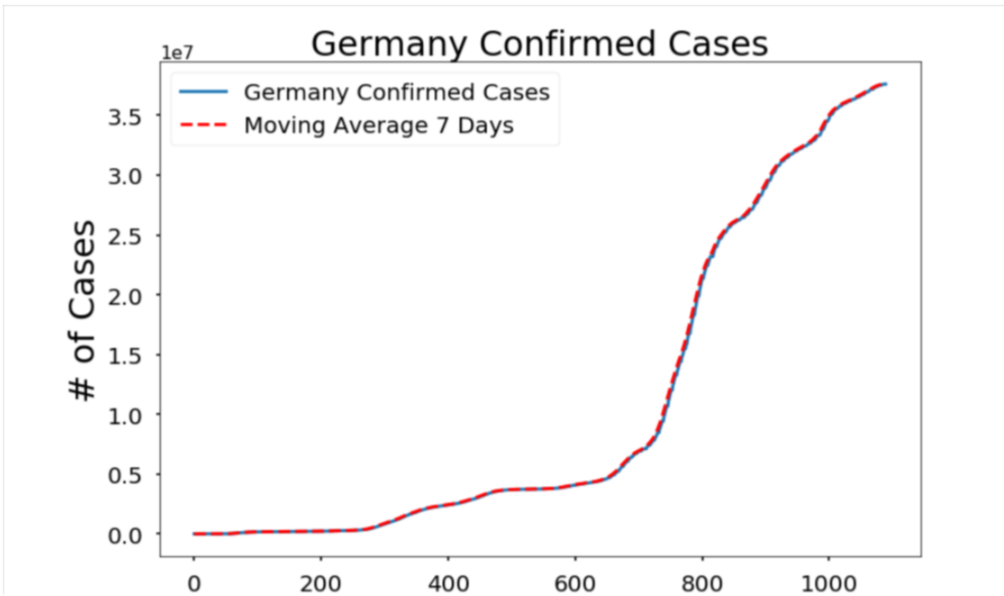
- **Germany**: Key findings for Germany's daily death and recovery data.

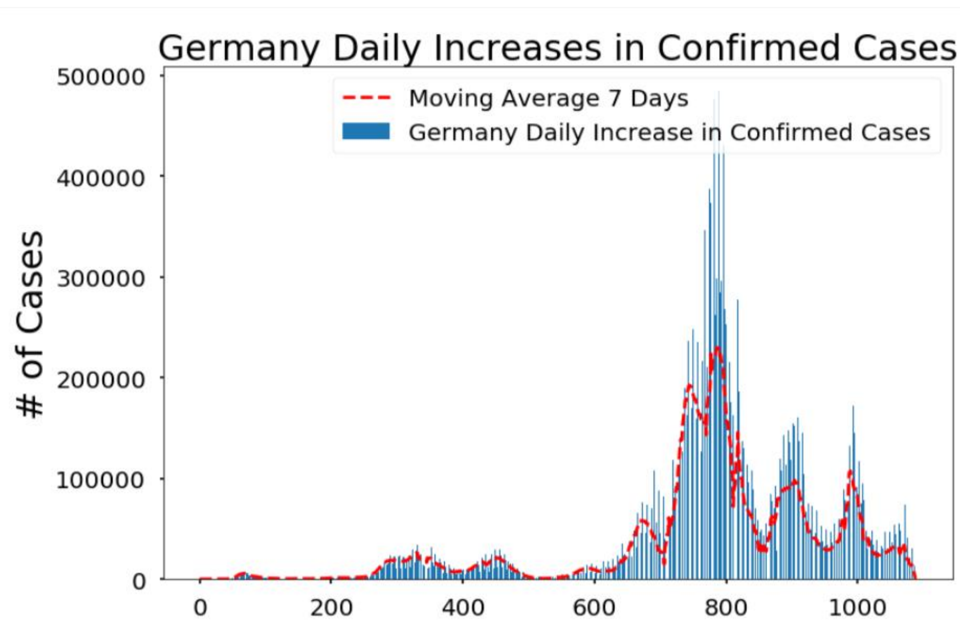
France: Key findings for **France's** daily death and recovery data.

- **Italy**: Key findings for Italy's daily death and recovery data.

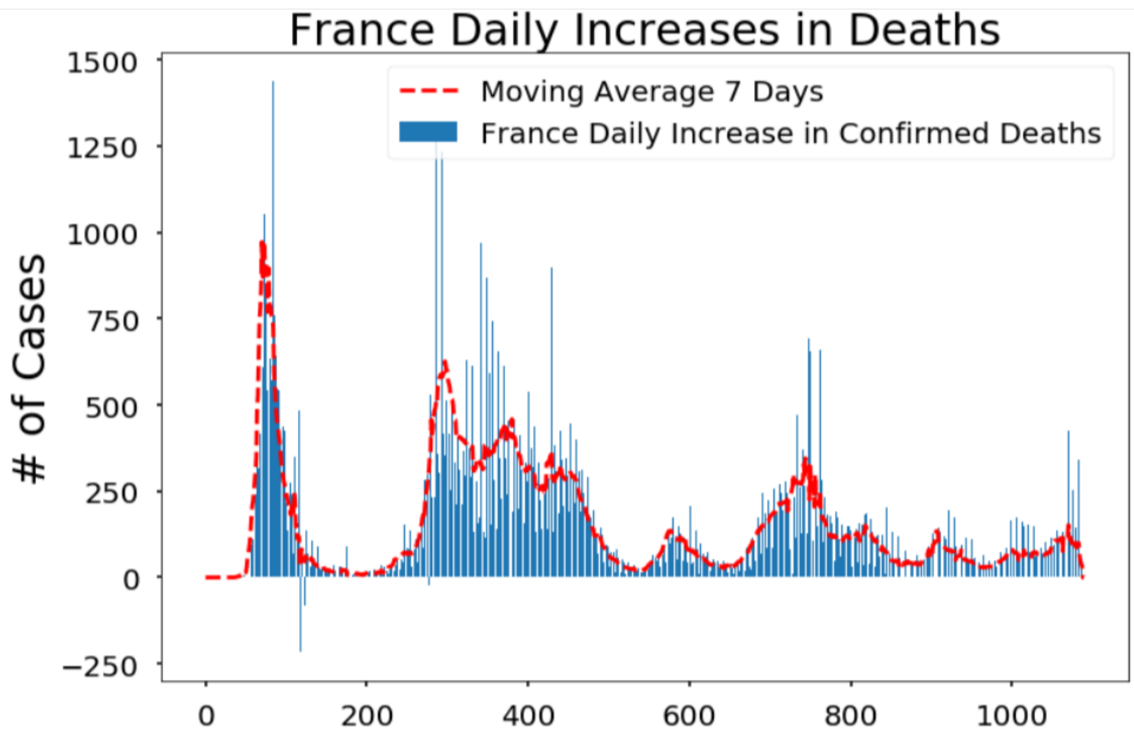
VISUALIZATIONS

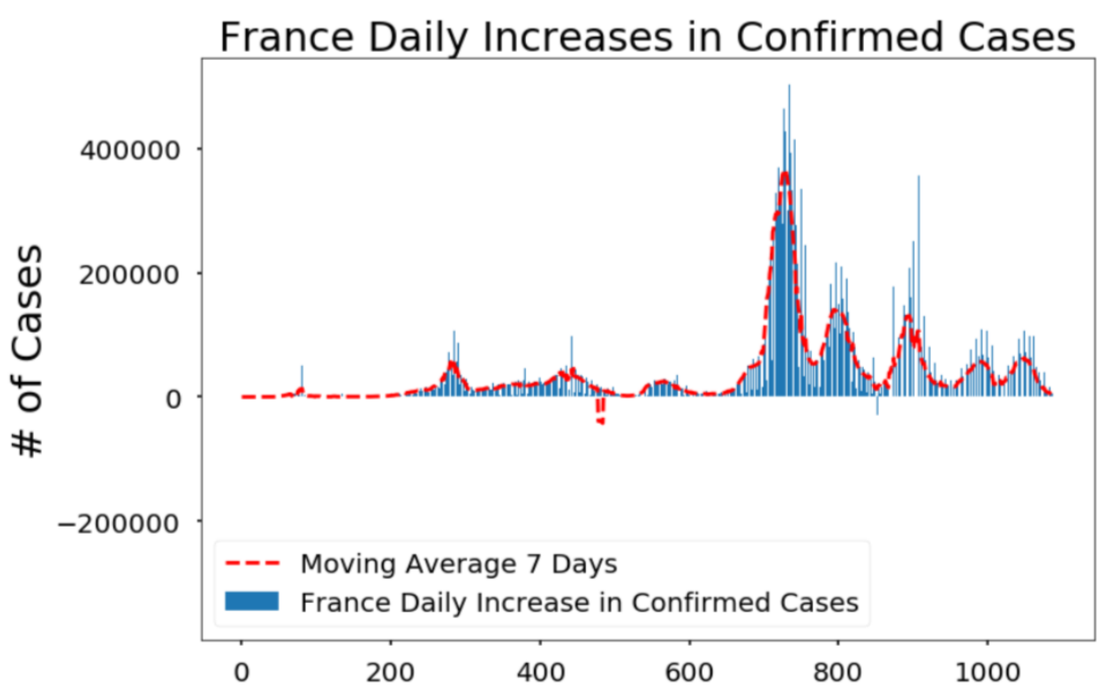
Germany:

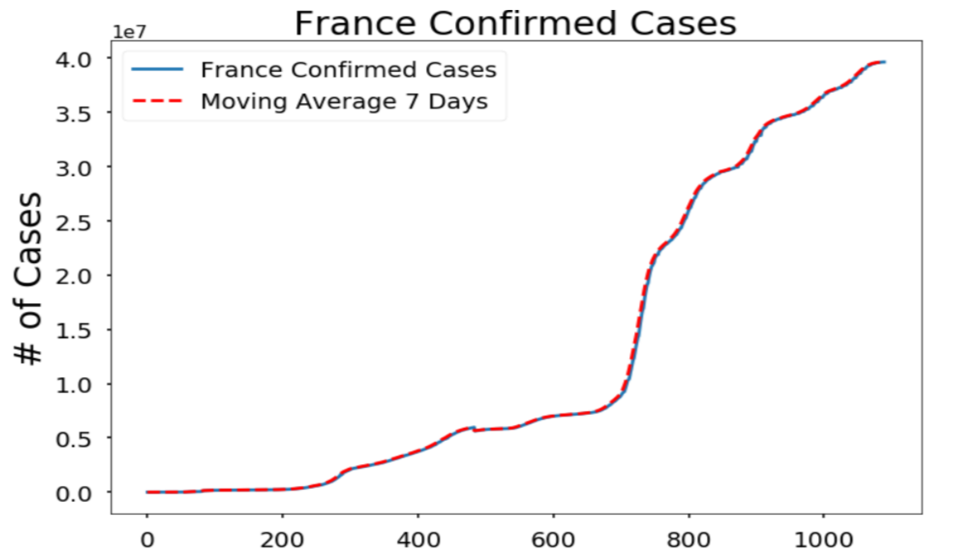




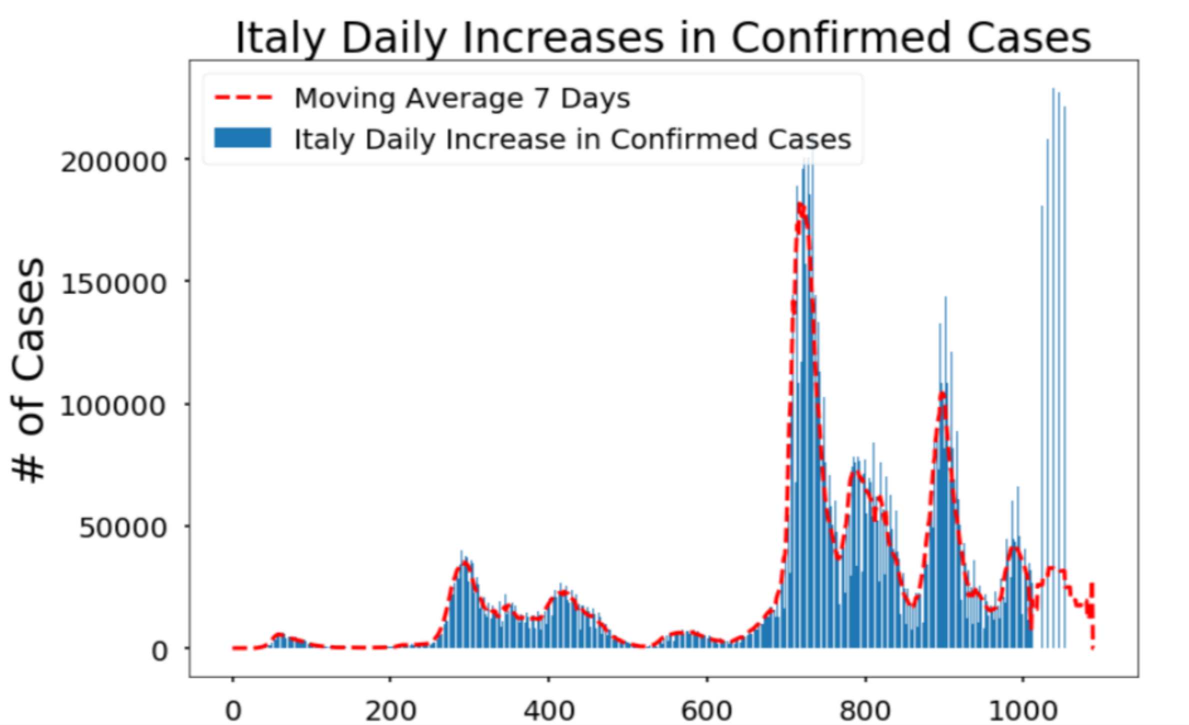
France:

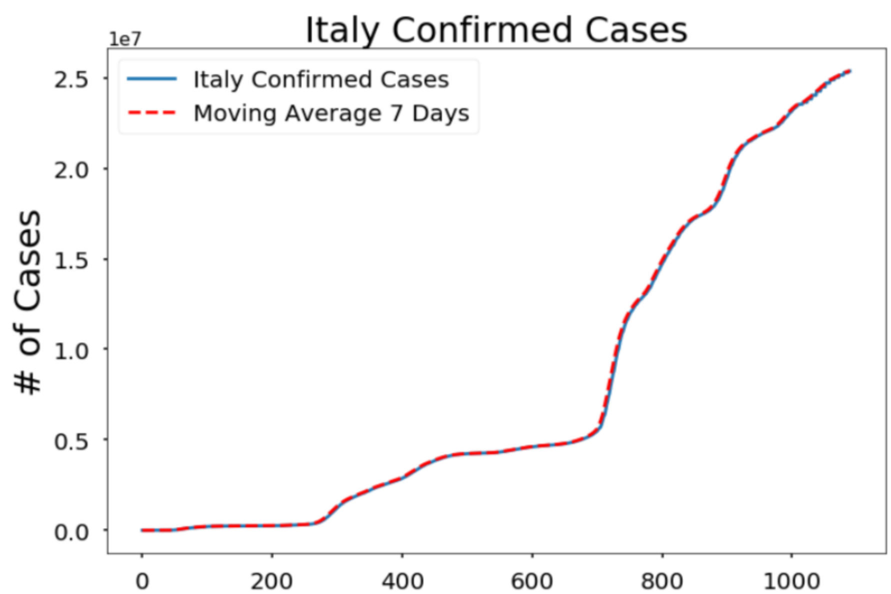

France Daily Increases in Deaths
Moving Average 7 Days
France Daily Increase in Confirmed Deaths
# of Cases
1500
1250
1000
750
500
250
0
−250
0
200
400
600
800
1000


France Daily Increases in Confirmed Cases
# of Cases
400000
200000
0
−200000
Moving Average 7 Days
France Daily Increase in Confirmed Cases
0
200
400
600
800
1000


France Confirmed Cases
1e7
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
# of Cases
France Confirmed Cases
Moving Average 7 Days
0
200
400
600
800
1000

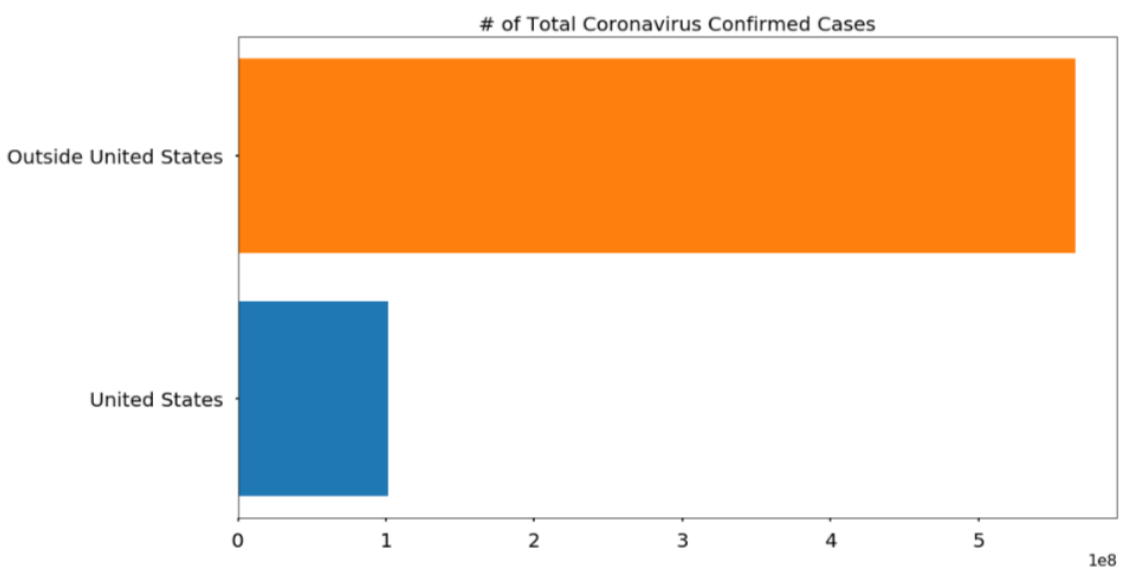
Italy


Italy Daily Increases in Confirmed Cases
Moving Average 7 Days
Italy Daily Increase in Confirmed Cases
200000
150000
100000
50000
0
# of Cases
0
200
400
600
800
1000

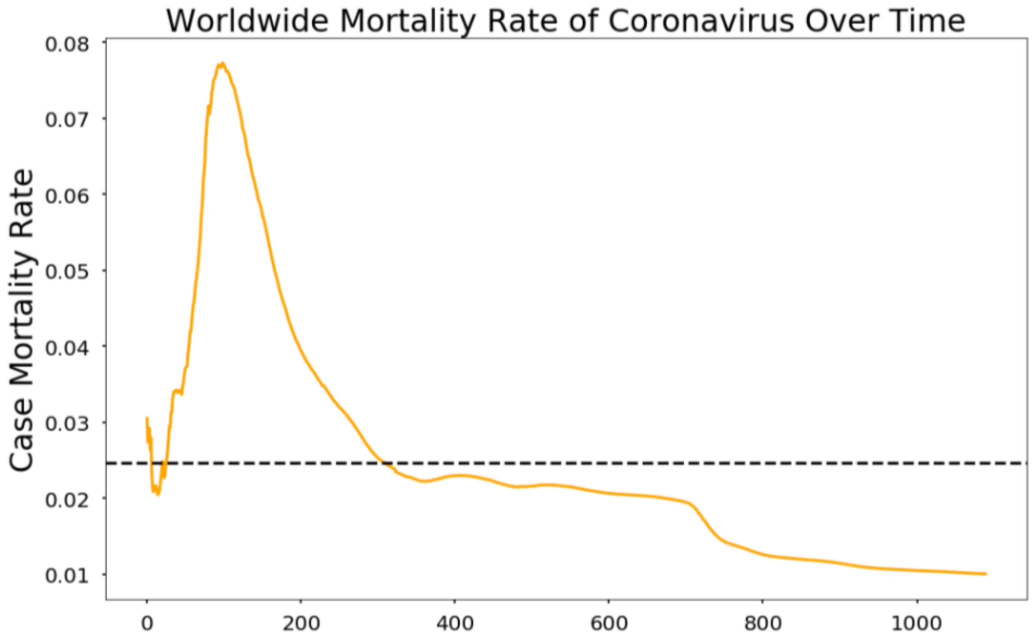


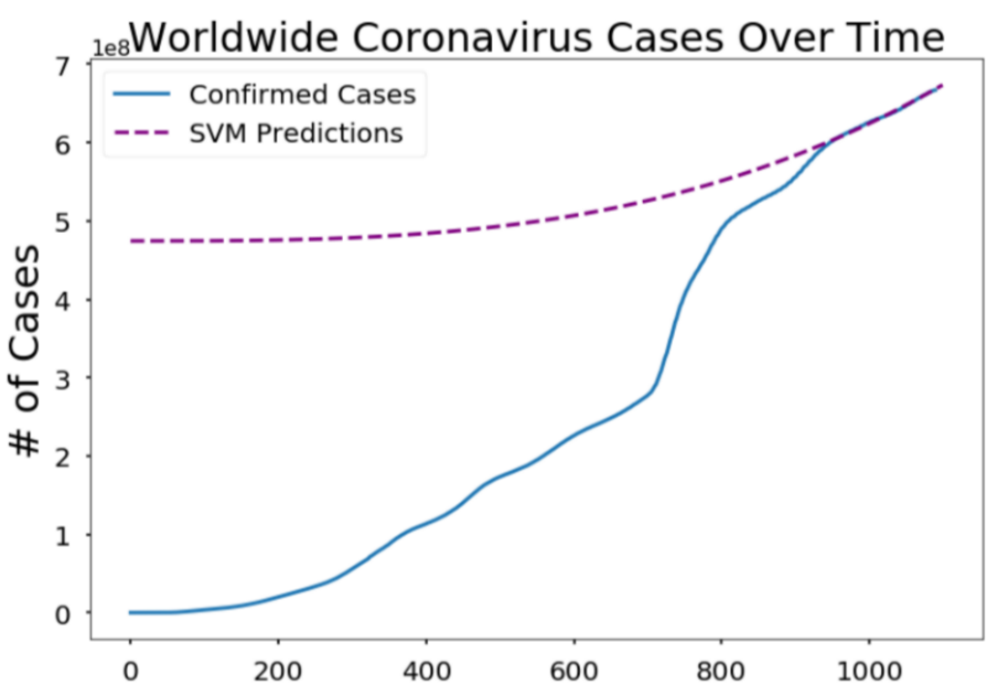
Italy Confirmed Cases
1e7
Italy Confirmed Cases
Moving Average 7 Days
# of Cases
2.5
2.0
1.5
1.0
0.5
0.0
0
200
400
600
800
1000

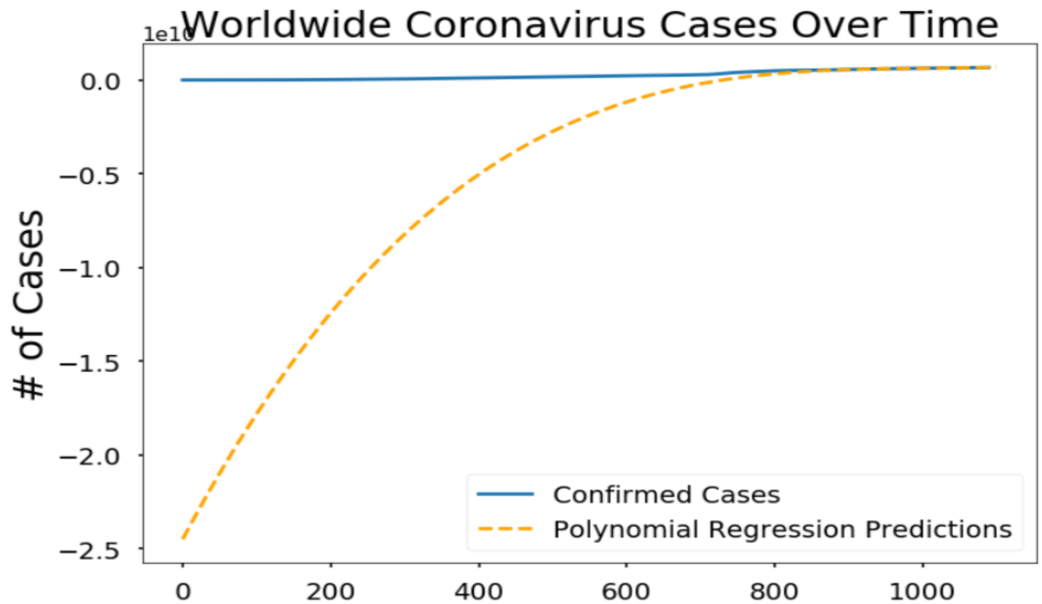
World trend:

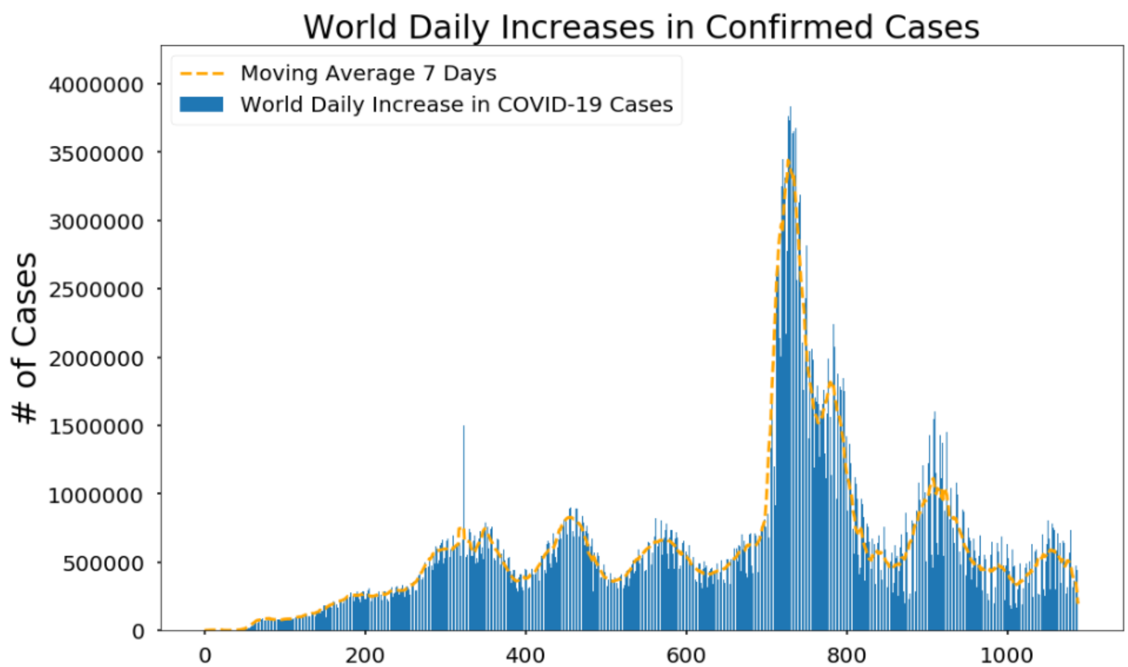


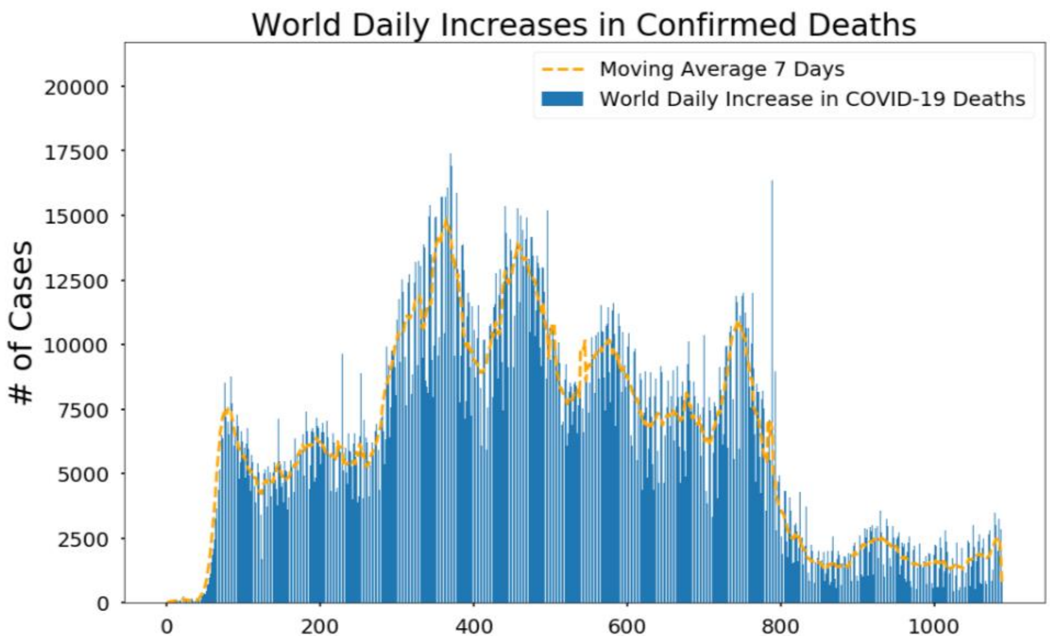
# of Total Coronavirus Confirmed Cases
Outside United States
United States
0
1
2
3
4
5
1e8

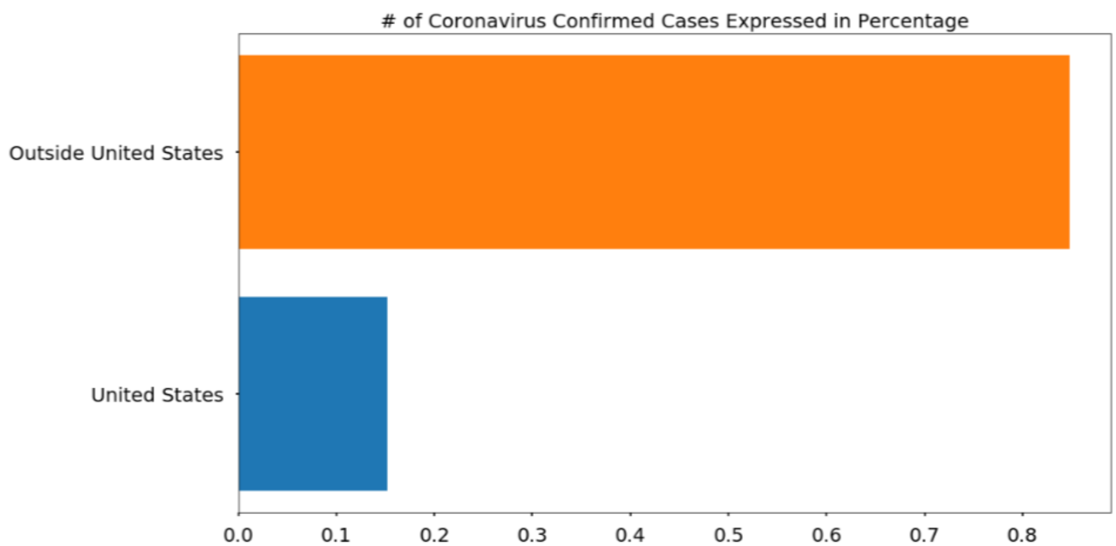

Worldwide Mortality Rate of Coronavirus Over Time
Case Mortality Rate
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
200
400
600
800
1000


Worldwide Coronavirus Cases Over Time
1e8
# of Cases
Confirmed Cases
SVM Predictions
7
6
5
4
3
2
1
0
0
200
400
600
800
1000


Worldwide Coronavirus Cases Over Time
1e10
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
# of Cases
0
200
400
600
800
1000
Confirmed Cases
Polynomial Regression Predictions


World Daily Increases in Confirmed Cases
Moving Average 7 Days
World Daily Increase in COVID-19 Cases
4000000
3500000
3000000
2500000
2000000
1500000
1000000
500000
0
# of Cases
0
200
400
600
800
1000


World Daily Increases in Confirmed Deaths
Moving Average 7 Days
World Daily Increase in COVID-19 Deaths
# of Cases
0
2500
5000
7500
10000
12500
15000
17500
20000
0
200
400
600
800
1000


# of Coronavirus Confirmed Cases Expressed in Percentage
Outside United States
United States
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8

PROGRAM:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.linear_model import LinearRegression, BayesianRidge
from sklearn.model_selection import RandomizedSearchCV, train_test_split
from sklearn.preprocessing import PolynomialFeatures
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, mean_absolute_error
import datetime

# Import and preprocess COVID-19 data
confirmed_df = pd.read_csv('confirmed_data_url')
deaths_df = pd.read_csv('deaths_data_url')
latest_data = pd.read_csv('latest_data_url')
us_medical_data = pd.read_csv('us_medical_data_url')

# Extract relevant columns
confirmed_cols = confirmed_df.keys()
deaths_cols = deaths_df.keys()
confirmed = confirmed_df.loc[:, confirmed_cols[4]:]
deaths = deaths_df.loc[:, deaths_cols[4]:]

# Analyze global COVID-19 data
```

```
num_dates = len(confirmed.keys())
ck = confirmed.keys()
dk = deaths.keys()
world_cases = []
total_deaths = []
mortality_rate = []

# Calculate total cases, deaths, and mortality rate
for i in range(num_dates):
    confirmed_sum = confirmed[ck[i]].sum()
    death_sum = deaths[dk[i]].sum()
    world_cases.append(confirmed_sum)
    total_deaths.append(death_sum)
    mortality_rate.append(death_sum / confirmed_sum)

# Define functions for data analysis and visualization
def daily_increase(data):
    # Calculate daily increase in data
    d = []
    for i in range(len(data)):
        if i == 0:
            d.append(data[0])
        else:
            d.append(data[i] - data[i-1])
    return d

def moving_average(data, window_size):
```

```
    # Calculate moving average of data
    moving_average = []
    for i in range(len(data)):
        if i + window_size < len(data):
            moving_average.append(np.mean(data[i:i+window_size]))
        else:
            moving_average.append(np.mean(data[i:len(data)]))
    return moving_average

# Specify window size for moving averages
window = 7

# Analyze and visualize COVID-19 cases and deaths
world_daily_increase = daily_increase(world_cases)
world_confirmed_avg = moving_average(world_cases, window)
world_daily_increase_avg = moving_average(world_daily_increase, window)
world_daily_death = daily_increase(total_deaths)
world_death_avg = moving_average(total_deaths, window)
world_daily_death_avg = moving_average(world_daily_death, window)

# Prepare data for regression modeling
days_since_1_22 = np.array([i for i in range(len(ck))]).reshape(-1, 1)
world_cases = np.array(world_cases).reshape(-1, 1)
total_deaths = np.array(total_deaths).reshape(-1, 1)
days_in_future = 10
future_forcast = np.array([i for i in range(len(ck) + days_in_future)]).reshape(-1, 1)
adjusted_dates = future_forcast[:-10]
```

```
start = '1/22/2020'

start_date = datetime.datetime.strptime(start, '%m/%d/%Y')

future_forcast_dates = []


# Generate future dates for forecasting

for i in range(len(future_forcast)):

    future_forcast_dates.append((start_date +
datetime.timedelta(days=i)).strftime('%m/%d/%Y'))


# Train and test data for regression models

days_to_skip = 830

X_train_confirmed, X_test_confirmed, y_train_confirmed, y_test_confirmed =
train_test_split(days_since_1_22[days_to_skip:], world_cases[days_to_skip:], test_size=0.10,
shuffle=False)


# Support Vector Regression (SVM) model for confirmed cases

svm_confirmed = SVR(shrinking=True, kernel='poly', gamma=0.01, epsilon=1, degree=3, C=0.1)

svm_confirmed.fit(X_train_confirmed, y_train_confirmed)

svm_pred = svm_confirmed.predict(future_forcast)


# Polynomial regression model for confirmed cases

poly = PolynomialFeatures(degree=3)

poly_X_train_confirmed = poly.fit_transform(X_train_confirmed)

poly_X_test_confirmed = poly.fit_transform(X_test_confirmed)

poly_future_forcast = poly.fit_transform(future_forcast)

linear_model = LinearRegression(normalize=True, fit_intercept=False)

linear_model.fit(poly_X_train_confirmed, y_train_confirmed)

test_linear_pred = linear_model.predict(poly_X_test_confirmed)
```

```
linear_pred = linear_model.predict(poly_future_forcast)

# Bayesian Ridge Polynomial Regression model for confirmed cases
tol = [1e-6, 1e-5, 1e-4, 1e-3, 1e-2]
alpha_1 = [1e-7, 1e-6, 1e-5, 1e-4, 1e-3]
alpha_2 = [1e-7, 1e-6, 1e-5, 1e-4, 1e-3]
lambda_1 = [1e-7, 1e-6, 1e-5, 1e-4, 1e-3]
lambda_2 = [1e-7, 1e-6, 1e-5, 1e-4, 1e-3]
normalize = [True, False]

bayesian_grid = {
    'tol': tol,
    'alpha_1': alpha_1,
    'alpha_2': alpha_2,
    'lambda_1': lambda_1,
    'lambda_2': lambda_2,
    'normalize': normalize
}

bayesian = BayesianRidge(fit_intercept=False)
bayesian_search = RandomizedSearchCV(bayesian, bayesian_grid,
scoring='neg_mean_squared_error', cv=3, return_train_score=True, n_jobs=-1, n_iter=40,
verbose=1)
bayesian_search.fit(bayesian_poly_X_train_confirmed, y_train_confirmed)
bayesian_confirmed = bayesian_search.best_estimator_
test_bayesian_pred = bayesian_confirmed.predict(bayesian_poly_X_test_confirmed)
bayesian_pred = bayesian_confirmed.predict(bayesian_poly_future_forcast)
```

```python
# Visualize top 10 total COVID-19 cases for specific countries
countries = ['US', 'India', 'Brazil', 'France', 'Germany', 'United Kingdom', 'Italy', 'Korea, South',
'Russia', 'Turkey']

for country in countries:
    country_visualizations(country)

# Compare COVID-19 cases and deaths in selected countries
compare_countries = ['India', 'US', 'Brazil', 'Russia', 'United Kingdom', 'France']
graph_name = ['Coronavirus Confirmed Cases', 'Coronavirus Confirmed Deaths']

for num in range(2):
    plt.figure(figsize=(12, 8))
    for country in compare_countries:
        plt.plot(get_country_info(country)[num])
    plt.legend(compare_countries, prop={'size': 20})
    plt.xlabel('Days since 1/22/2020', size=30)
    plt.ylabel('# of Cases', size=30)
    plt.title(graph_name[num], size=30)
    plt.xticks(size=20)
    plt.yticks(size=20)
    plt.show()
```

Conclusion

The analysis of daily death and recovery data for Germany, France, and Italy provides valuable insights into the COVID-19 pandemic's impact in these countries. Our findings offer information that can guide policy decisions, healthcare resource allocation, and public health measures.