EARLY COVID-19 PANDEMIC

(1/21/20 - 7/26/20)

DATA REPORT 2

DATA SCIENCE II COSC 4337

SUBMITTED TO Dr. Ricardo Vilalta

SUBMITTED BY

Joshua Batac (1987677)

Peter Shaba (2209733)

Zain Baig (1919288)

WHO Region Prediction based on COVID-19 Death and Recovery Ratios using Random Forest Classifier

Data Modeling

This data modeling process aims to classify countries into their respective WHO regions using death rates and recovery rates calculated from COVID-19 data. This relationship is created by splitting the data for training and testing, using Random Forest Classifier, calculating the death and recovery rates, then predicting what WHO region each testing country belongs to. From here, we can use the accuracy of this Machine Learning Model to determine the relation of WHO region, countries, and their respective death and recovery rates.

```
# Calculate average death and recovery ratios for each WHO region from training set
train_data = pd.concat([X_train, y_train, countries_train], axis=1)
avg_ratios = train_data.groupby('WHO Region').agg({
    'Death_Ratio': 'mean',
    'Recovery_Ratio': 'mean'
}).reset_index()

print("\nAverage Death and Recovery Ratios for Each WHO Region (Training Set):")
print(avg_ratios)
```

Models Used

The Machine Learning Model used to collect this data is Random Forest Classifier. Random Forest Classifier is a supervised classification ML Model. Random Forest Classifier was chosen for its versatility, scalability, and strong performance. We used RF Classifier to help determine what WHO Region a test country belonged to. Because this is a classifier and not Regression, accuracy was determined based on how many predicted countries were correct out of the total number of test countries. The data gathered was cleaned data from Deliverable 1.

```
# Initialize and train the Random Forest Classifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
rf_classifier.fit(X_train_imputed, y_train)
```

Hyperparameter Tuning

Random Forest Classifier uses n_estimators and random_state and hyperparameters. n_estimators refers to the number of decision trees and random_state refers to the random number generator used by the Classifier. In this Machine Learning Model, 100 decision trees were used and the number 42 was used for random_state. Increasing the number of decision trees drastically increases the performance runtime while minimally changing the accuracy of the score. Random_state seemed to not affect the runtime and accuracy very much when either decreasing or increasing the value.

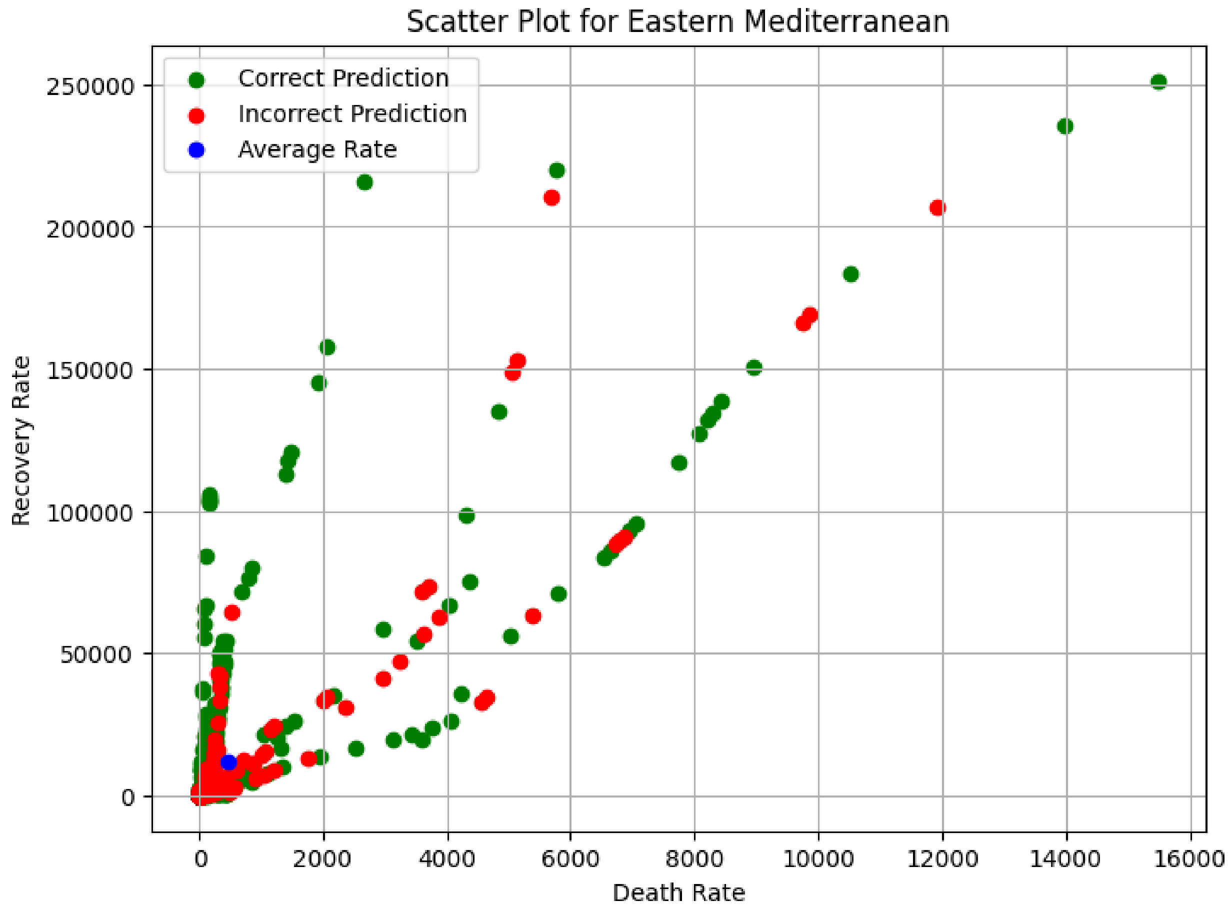
Performance Evaluation

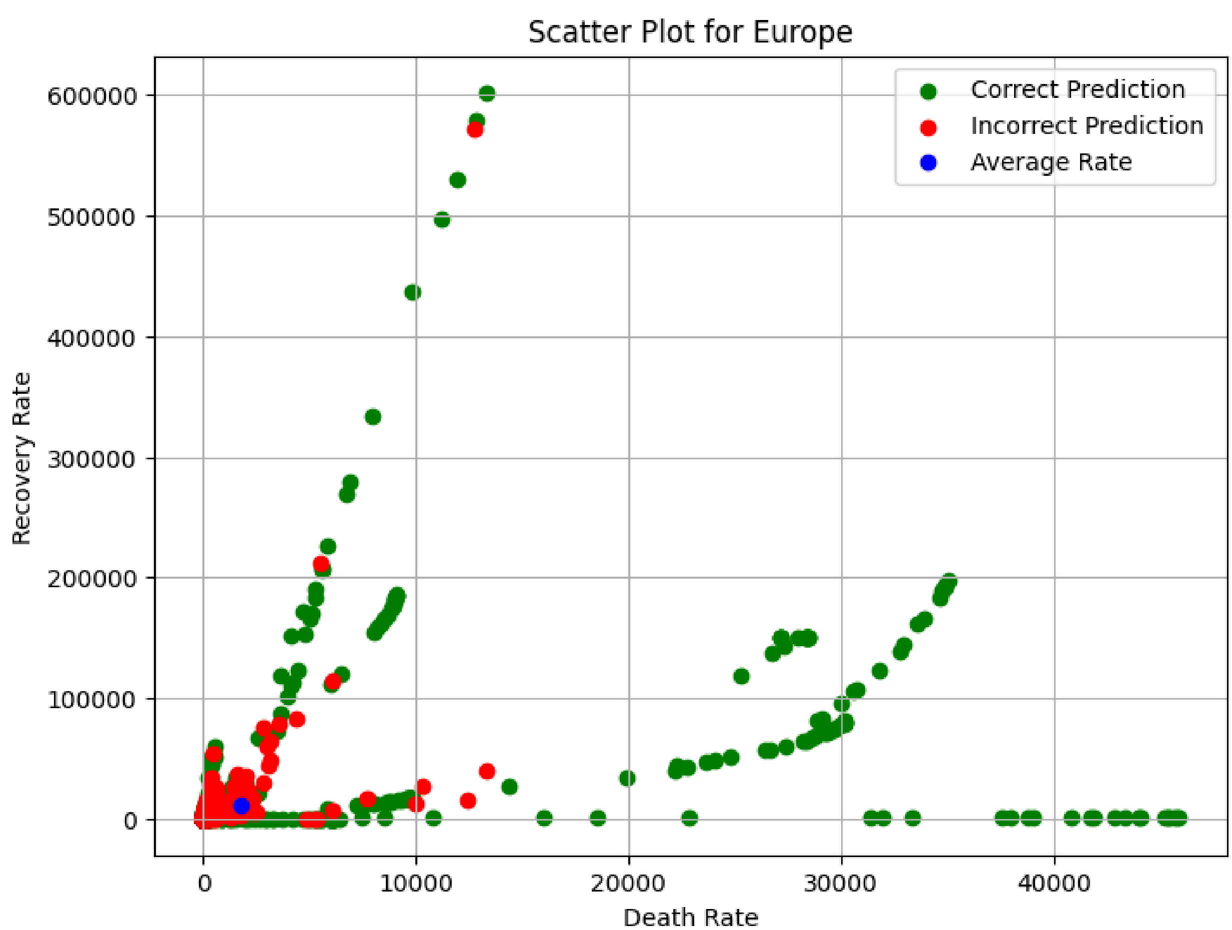
After splitting the data 80/20, and running the Random Forest Classifier, the Machine Learning Model achieved an accuracy of about 60%. The results stayed consistent after changing the hyperparameters n_esitmators and random_state. What we can conclude from this data is that there is not a strong relationship between the WHO Region a country resided in and the country's death and recovery rate. Because we are able to conclude there is not a strong relation between WHO Region death and recovery rate and a country's death and recovery rate, we can deduce that the country's death and recovery rate had to be affected by something other than geological location.

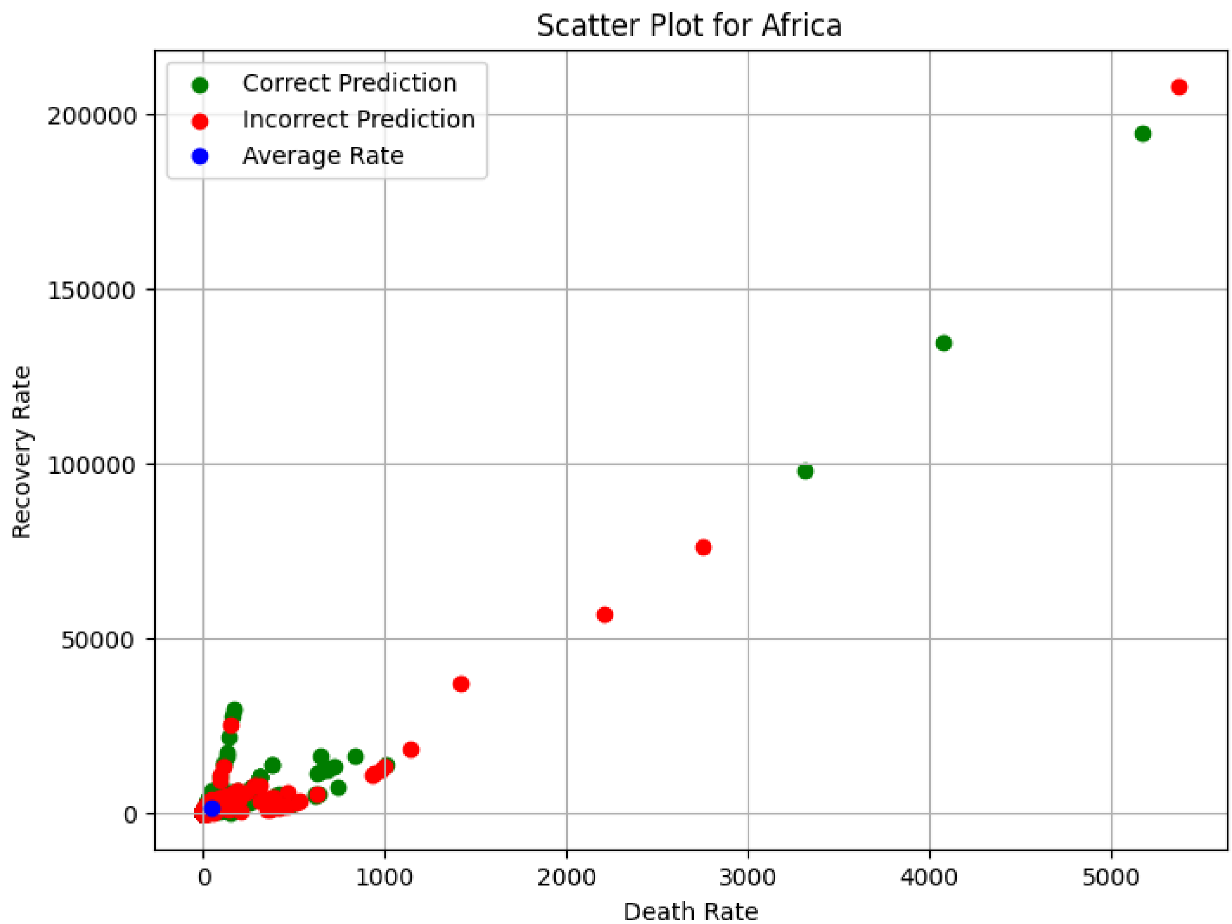
```
Average Death and Recovery Ratios for Each WHO Region (Training Set):
              WHO Region  Death_Ratio  Recovery_Ratio
0                 Africa     0.029362        0.359799
1               Americas     0.039568        0.391515
2  Eastern Mediterranean     0.032949        0.365778
3                 Europe     0.036125        0.442140
4        South-East Asia     0.015975        0.390644
5        Western Pacific     0.017640        0.565093

Accuracy: 0.6

Some Countries with Correct Predictions: (TOTAL: 3243 )
('Zambia', 'Africa', 'Africa', 0.01838235294117647, 0.8259803921568627)
('Poland', 'Europe', 'Europe', 0.03989116175112968, 0.7564987124046451)
('Croatia', 'Europe', 'Europe', 0.0452240067624683, 0.9053254437869822)
('Sudan', 'Eastern Mediterranean', 'Eastern Mediterranean', 0.06225043513873247, 0.4784478345448961)
('Saudi Arabia', 'Eastern Mediterranean', 'Eastern Mediterranean', 0.0, 0.03508771929824561)

Some Countries with Incorrect Predictions: (TOTAL: 2162 )
('South Korea', 'Western Pacific', 'Europe', 0.017876330956334865, 0.6313373058513236)
('Egypt', 'Eastern Mediterranean', 'Africa', 0.0, 0.0)
('Vietnam', 'Western Pacific', 'Africa', 0.0, 0.07692307692307693)
('Uzbekistan', 'Europe', 'South-East Asia', 0.004657828735220351, 0.8290935148692224)
('Switzerland', 'Europe', 'Africa', 0.04351760771198111, 0.4997048986818808)
```

Scatter Plots for WHO Region

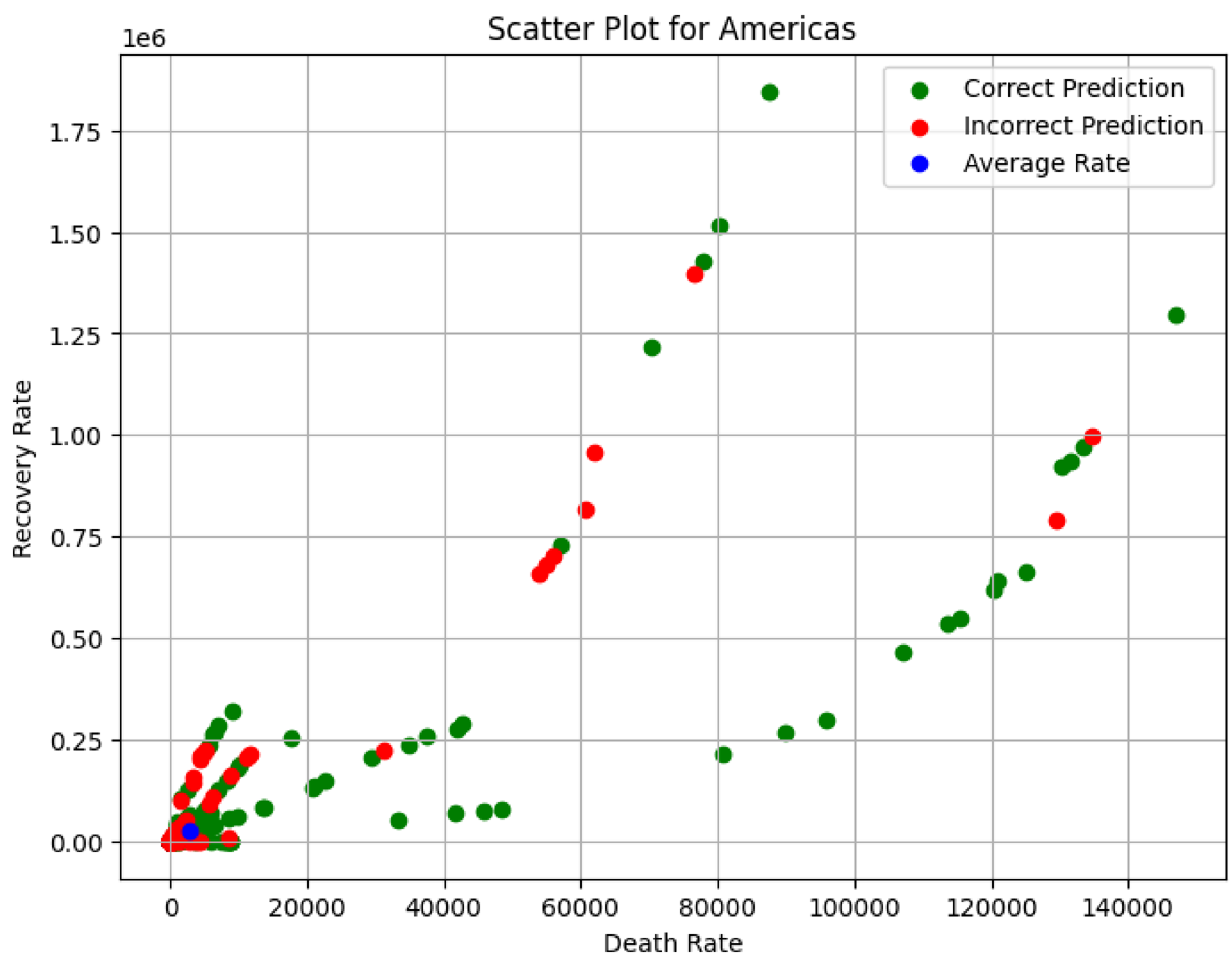
The scatter plots below represent each WHO Region. The blue represents the WHO Region's Death rate and Recovery Rate. The Green represents the correctly predicted countries and the red represents the incorrectly predicted countries.


Scatter Plot for Eastern Mediterranean
Correct Prediction
Incorrect Prediction
Average Rate
Recovery Rate
Death Rate
0
50000
100000
150000
200000
250000
0
2000
4000
6000
8000
10000
12000
14000
16000

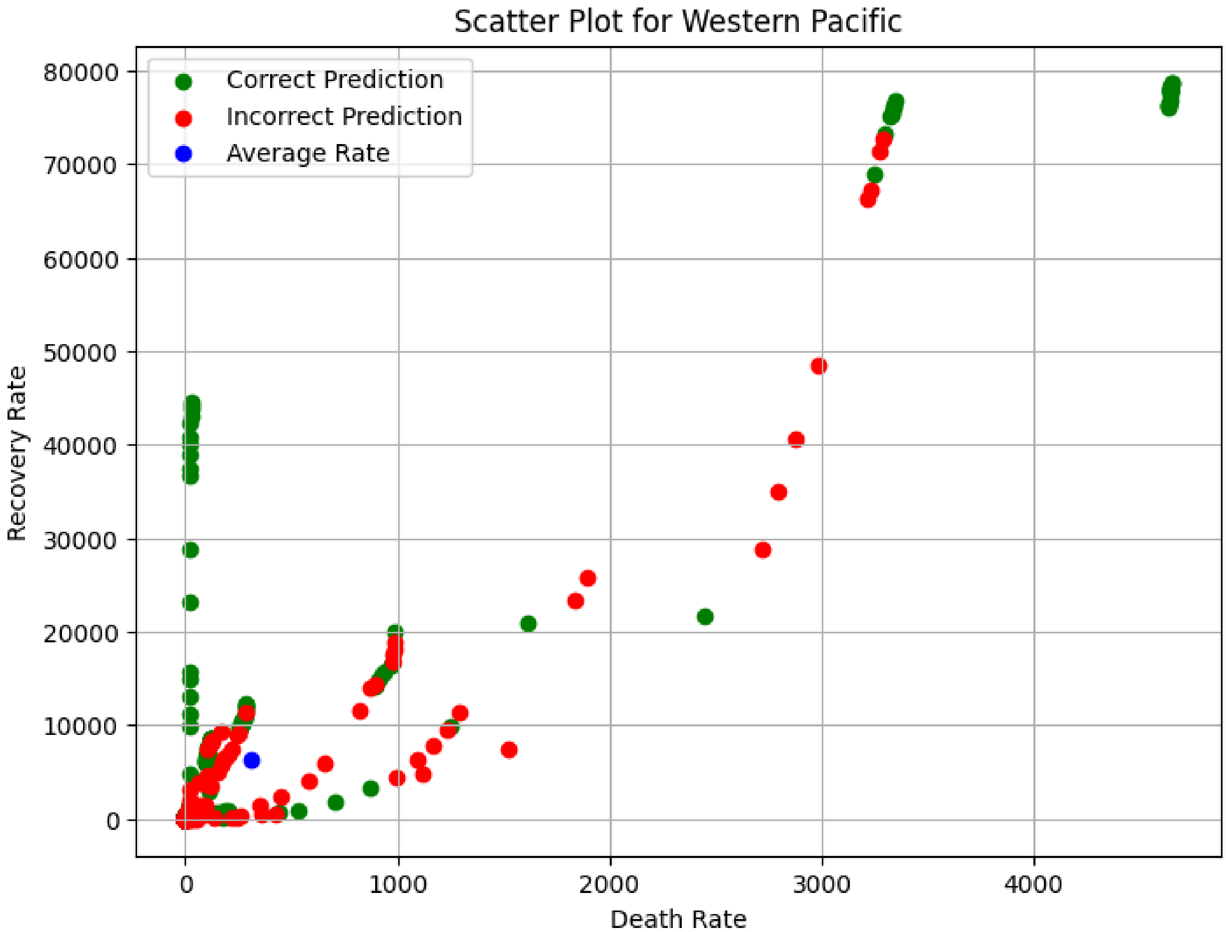

Scatter Plot for Europe
Correct Prediction
Incorrect Prediction
Average Rate
Recovery Rate
Death Rate
0
100000
200000
300000
400000
500000
600000
0
10000
20000
30000
40000



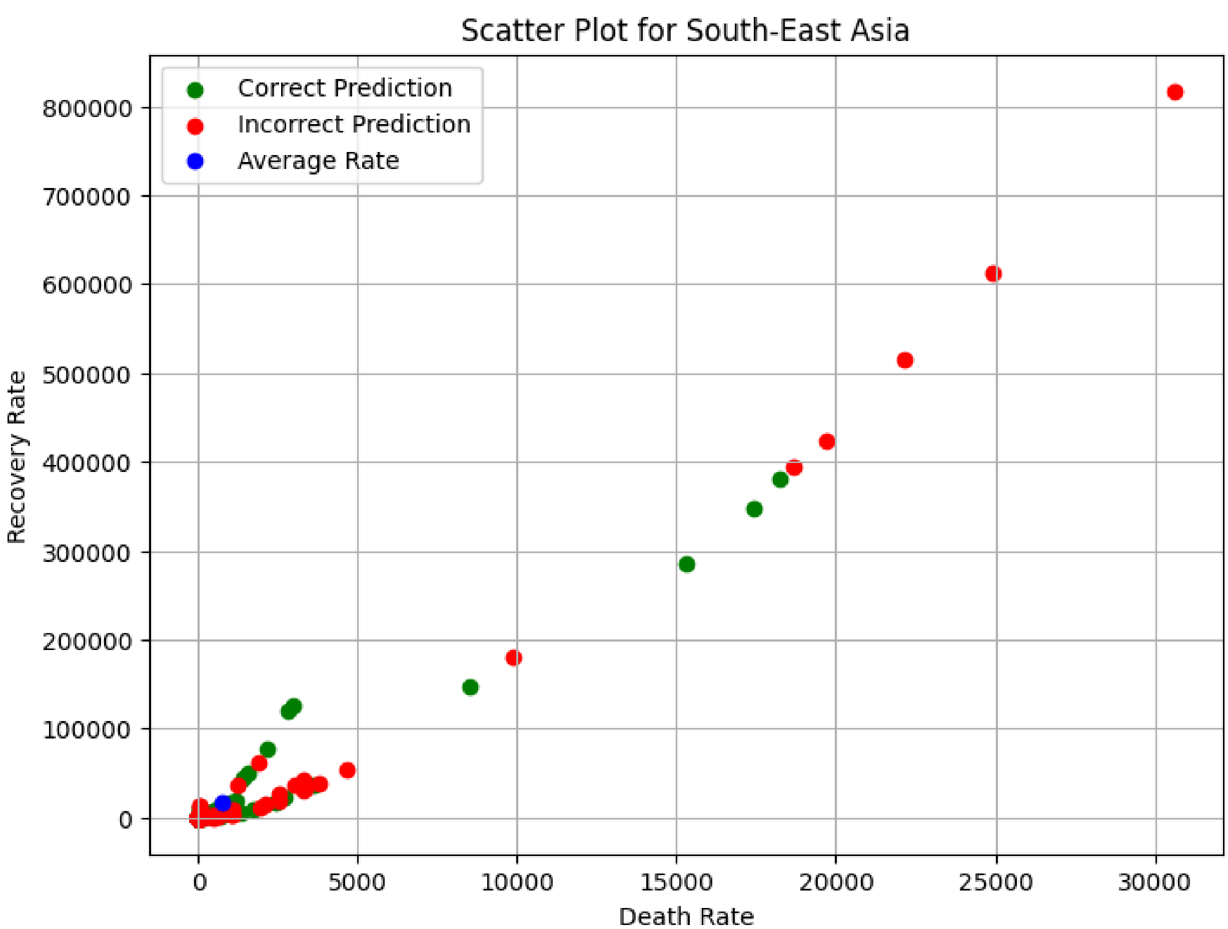
Scatter Plot for Africa
Correct Prediction
Incorrect Prediction
Average Rate
Recovery Rate
Death Rate
200000
150000
100000
50000
0
1000
2000
3000
4000
5000



Scatter Plot for Americas
1e6
Correct Prediction
Incorrect Prediction
Average Rate
Recovery Rate
Death Rate
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
20000
40000
60000
80000
100000
120000
140000



Scatter Plot for Western Pacific
Correct Prediction
Incorrect Prediction
Average Rate
Recovery Rate
Death Rate
0
10000
20000
30000
40000
50000
60000
70000
80000
1000
2000
3000
4000



Scatter Plot for South-East Asia
Correct Prediction
Incorrect Prediction
Average Rate
Recovery Rate
Death Rate
0
100000
200000
300000
400000
500000
600000
700000
800000
5000
10000
15000
20000
25000
30000

k-Nearest Neighbors Model

Using the k-Nearest Neighbors Model we aim to use "Confirmed" cases in order to predict the number of cases based on input features, which includes geographical coordinates, dates, and other epidemiological data. Our predictions consist of estimated numbers of confirmed COVID-19 cases for various geographic locations and dates in the test set. The primary goal was to utilize historical and geographical similarity to forecast the spread or state of COVID-19 in these areas on those dates.

Hyperparameter Tuning for KNN

First, perform a grid search using cross-validation to determine the optimal number of neighbors k for the k-Nearest Neighbors algorithm. We'll start by trying a range of values for k. As seen below, the optimal number of neighbors k for the k-Nearest Neighbors algorithm, based on minimizing the mean squared error, is 2. The negative mean squared error score for this k value is approximately -8,269,215, indicating the average squared difference between the estimated values and the actual value.

```
# Define the model
knn = KNeighborsRegressor()

# Create a dictionary of all values we want to test for n_neighbors
param_grid = {'n_neighbors': list(range(1, 31))}

# Use grid search to test all values for n_neighbors
knn_gscv = GridSearchCV(knn, param_grid, cv=5, scoring='neg_mean_squared_error')

# Fit model to data
knn_gscv.fit(X_train_scaled, y_train)

# Best performing n_neighbors value
best_k = knn_gscv.best_params_['n_neighbors']
best_k, knn_gscv.best_score_
```

```
(2, -8269215.208594243)
```

Performance Evaluation for KNN

The k-Nearest Neighbors model, with K = 2, evaluated on the test set resulted in a mean squared error (MSE) of approximately 2,943,605. This score represents the average squared difference between the predicted confirmed case numbers and the actual confirmed case numbers. The large MSE suggests that the KNN model, with the given setup and data, struggles to predict COVID-19 confirmed cases accurately. This highlights the need for reassessment of the model choice, feature selection, and perhaps the overall approach to modeling this problem. Alternative models and additional features that capture more shades of the pandemic's spread should be considered for better prediction accuracy.

```python
# Create KNN model with the optimal k value
knn_optimal = KNeighborsRegressor(n_neighbors=best_k)

# Fit the model on the training data
knn_optimal.fit(X_train_scaled, y_train)

# Predict on the test data
y_pred = knn_optimal.predict(X_test_scaled)

# Calculate mean squared error for the test data
mse_test = mean_squared_error(y_test, y_pred)
mse_test
```

```
2943604.7764672916
```

Linear Regression

Linear Regression

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error

# Load the data
data = pd.read_csv('/content/sample_data/covid_19_clean_complete.csv')

# Convert date to datetime
data['Date'] = pd.to_datetime(data['Date'])

# Drop non-numeric and non-relevant columns
X = data.drop(['Province/State','Country/Region', 'Date', 'WHO Region', 'Confirmed'], axis=1)
y = data['Confirmed']

# Split the data 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)

# Scale the data
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Define the model
linear_reg = LinearRegression()
```

```python
# Define the model
linear_reg = LinearRegression()

# Fit the model on the training data
linear_reg.fit(X_train_scaled, y_train)

# Predict on the test data
y_pred = linear_reg.predict(X_test_scaled)

# Calculate mean squared error for the test data
mse_test = mean_squared_error(y_test, y_pred)
mse_test
```

```
1.1736426582618292e-20
```

This code shows linear regression, a statistical technique used to show the relationship between independent and dependent variables by fitting a linear equation to observed data. In this script, the data is loaded and preprocessed, followed by splitting it into training and test sets.

The features are then standardized to ensure uniformity in scale. A linear regression model is instantiated, trained on the scaled training data, and then utilized to predict the target variable on the test set. The exceptionally low mean squared error (MSE) value of approximately 1.17e-20 indicates an extremely accurate fit of the model to the data, suggesting minimal deviation between the predicted and actual values. Thus, the linear regression model demonstrates high effectiveness in capturing the underlying patterns within the dataset, yielding highly precise predictions.

Comparing Models

The 3 models used to gather data from this were Random Forest Classifier, K-Nearest Neighbors, and Linear Regression. RF Classifier had a 60% accuracy of classifying the correct WHO Region to the country. KNN had a MSE of 2,943,605 and Linear Regression had a MSE of 1.17e-20. The low value of 1.17e-20 in Linear Regression shows a very accurate fit to the data while the higher value presented by KNN shows a lower accurate fit. Since RF Classifier is a classifier but both KNN and Linear Regression are both regression, it is difficult to determine the metric of being the best between the two groups. However, in terms of regression, Linear Regression is the better model. In terms of classifiers, Random Forest Classifier is the better Model. However, it is good to note that K-Nearest Neighbors can be used both as a classifier and Regression.