

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_estimators` 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	0.442148
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)
```

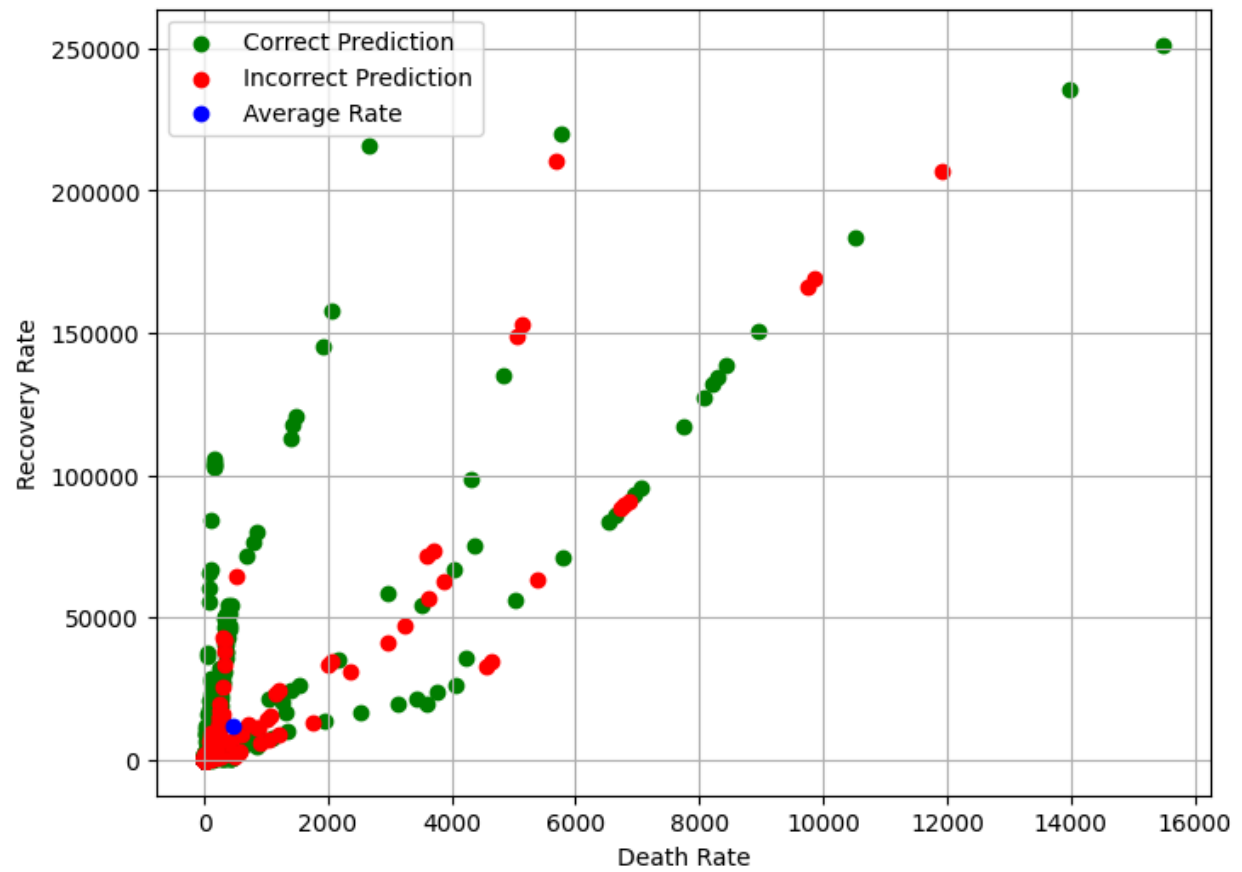
```
Some Countries with Incorrect Predictions: (TOTAL: 2162 )
```

```
('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)
```

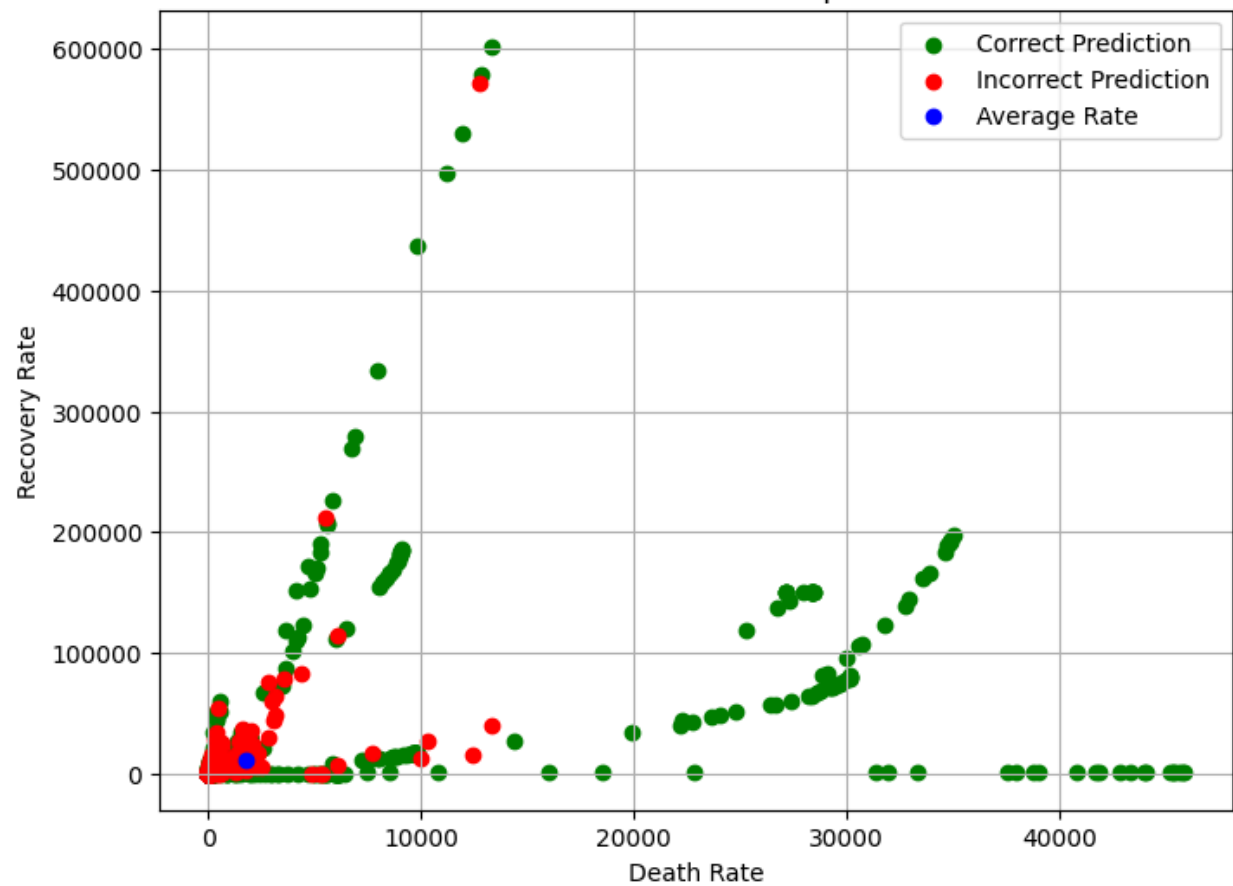
## Scatter Plots for WHO Region

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



Scatter Plot for Europe



A scatter plot showing the relationship between the Death Rate (X-axis) and the Recovery Rate (Y-axis). The X-axis ranges from 0 to 5000, and the Y-axis ranges from 0 to 200,000. The plot compares two categories: Correct Prediction (green dots) and Incorrect Prediction (red dots). A legend in the top-left corner identifies these categories. The data points for both categories are clustered at low Death Rates (below 1000) and low Recovery Rates (below 50,000). There are a few outliers for both categories at higher Death Rates and Recovery Rates. Specifically, there are green dots at approximately (3300, 100000), (4100, 135000), (5200, 195000), and (5400, 210000). There are red dots at approximately (2200, 55000), (2700, 75000), (1400, 35000), and (5400, 210000).

Death Rate	Recovery Rate	Prediction Type
0	0	Correct Prediction
0	0	Incorrect Prediction
100	10000	Correct Prediction
100	20000	Correct Prediction
100	30000	Correct Prediction
100	25000	Incorrect Prediction
200	15000	Correct Prediction
200	25000	Correct Prediction
200	30000	Incorrect Prediction
300	10000	Correct Prediction
300	15000	Correct Prediction
300	20000	Correct Prediction
300	25000	Correct Prediction
300	30000	Correct Prediction
300	35000	Correct Prediction
300	40000	Correct Prediction
300	45000	Correct Prediction
300	50000	Correct Prediction
300	55000	Correct Prediction
300	60000	Correct Prediction
300	65000	Correct Prediction
300	70000	Correct Prediction
300	75000	Correct Prediction
300	80000	Correct Prediction
300	85000	Correct Prediction
300	90000	Correct Prediction
300	95000	Correct Prediction
300	100000	Correct Prediction
300	105000	Correct Prediction
300	110000	Correct Prediction
300	115000	Correct Prediction
300	120000	Correct Prediction
300	125000	Correct Prediction
300	130000	Correct Prediction
300	135000	Correct Prediction
300	140000	Correct Prediction
300	145000	Correct Prediction
300	150000	Correct Prediction
300	155000	Correct Prediction
300	160000	Correct Prediction
300	165000	Correct Prediction
300	170000	Correct Prediction
300	175000	Correct Prediction
300	180000	Correct Prediction
300	185000	Correct Prediction
300	190000	Correct Prediction
300	195000	Correct Prediction
300	200000	Correct Prediction
300	205000	Correct Prediction
300	210000	Correct Prediction
300	215000	Correct Prediction
300	220000	Correct Prediction
300	225000	Correct Prediction
300	230000	Correct Prediction
300	235000	Correct Prediction
300	240000	Correct Prediction
300	245000	Correct Prediction
300	250000	Correct Prediction
300	255000	Correct Prediction
300	260000	Correct Prediction
300	265000	Correct Prediction
300	270000	Correct Prediction
300	275000	Correct Prediction
300	280000	Correct Prediction
300	285000	Correct Prediction
300	290000	Correct Prediction
300	295000	Correct Prediction
300	300000	Correct Prediction
300	305000	Correct Prediction
300	310000	Correct Prediction
300	315000	Correct Prediction
300	320000	Correct Prediction
300	325000	Correct Prediction
300	330000	Correct Prediction
300	335000	Correct Prediction
300	340000	Correct Prediction
300	345000	Correct Prediction
300	350000	Correct Prediction
300	355000	Correct Prediction
300	360000	Correct Prediction
300	365000	Correct Prediction
300	370000	Correct Prediction
300	375000	Correct Prediction
300	380000	Correct Prediction
300	385000	Correct Prediction
300	390000	Correct Prediction
300	395000	Correct Prediction
300	400000	Correct Prediction
300	405000	Correct Prediction
300	410000	Correct Prediction
300	415000	Correct Prediction
300	420000	Correct Prediction
300	425000	Correct Prediction
300	430000	Correct Prediction
300	435000	Correct Prediction
300	440000	Correct Prediction
300	445000	Correct Prediction
300	450000	Correct Prediction
300	455000	Correct Prediction
300	460000	Correct Prediction
300	465000	Correct Prediction
300	470000	Correct Prediction
300	475000	Correct Prediction
300	480000	Correct Prediction
300	485000	Correct Prediction
300	490000	Correct Prediction
300	495000	Correct Prediction
300	500000	Correct Prediction
300	505000	Correct Prediction
300	510000	Correct Prediction
300	515000	Correct Prediction
300	520000	Correct Prediction
300	525000	Correct Prediction
300	530000	Correct Prediction
300	535000	Correct Prediction
300	540000	Correct Prediction
300	545000	Correct Prediction
300	550000	Correct Prediction
300	555000	Correct Prediction
300	560000	Correct Prediction
300	565000	Correct Prediction
300	570000	Correct Prediction
300	575000	Correct Prediction
300	580000	Correct Prediction
300	585000	Correct Prediction
300	590000	Correct Prediction
300	595000	Correct Prediction

A scatter plot showing the relationship between the Death Rate (X-axis) and the Recovery Rate (Y-axis) for three different prediction methods. The X-axis ranges from 0 to 4500, and the Y-axis ranges from 0 to 80,000. The legend indicates three data series: Correct Prediction (green dots), Incorrect Prediction (red dots), and Average Rate (blue dots). The plot shows a general trend where higher death rates are associated with higher recovery rates, particularly for the 'Correct Prediction' series. The 'Incorrect Prediction' series shows a more scattered distribution, often with lower recovery rates for similar death rates. The 'Average Rate' is represented by a single blue dot at a low death rate and low recovery rate.

A scatter plot showing the relationship between the Death Rate (X-axis) and the Recovery Rate (Y-axis). The X-axis ranges from 0 to 30,000, and the Y-axis ranges from 0 to 800,000. The plot compares two sets of data: 'Correct Prediction' (green dots) and 'Incorrect Prediction' (red dots). A legend in the top-left corner identifies these series. The 'Correct Prediction' series shows a positive correlation, with points generally following a linear trend from the origin up to a death rate of about 10,000. The 'Incorrect Prediction' series shows a much steeper positive correlation, with points starting at higher recovery rates for the same death rates and continuing to rise sharply as the death rate increases beyond 10,000. A single blue dot, labeled 'Average Rate' in the legend, is located near the origin at approximately (1,000, 20,000).

Death Rate	Recovery Rate	Prediction Type
0	0	Correct Prediction
1000	20000	Average Rate
1000	50000	Correct Prediction
2000	100000	Correct Prediction
3000	150000	Correct Prediction
4000	200000	Correct Prediction
5000	250000	Correct Prediction
6000	300000	Correct Prediction
7000	350000	Correct Prediction
8000	400000	Correct Prediction
9000	450000	Correct Prediction
10000	500000	Correct Prediction
11000	550000	Correct Prediction
12000	600000	Correct Prediction
13000	650000	Correct Prediction
14000	700000	Correct Prediction
15000	750000	Correct Prediction
16000	800000	Correct Prediction
17000	850000	Correct Prediction
18000	900000	Correct Prediction
19000	950000	Correct Prediction
20000	1000000	Correct Prediction
21000	1050000	Correct Prediction
22000	1100000	Correct Prediction
23000	1150000	Correct Prediction
24000	1200000	Correct Prediction
25000	1250000	Correct Prediction
26000	1300000	Correct Prediction
27000	1350000	Correct Prediction
28000	1400000	Correct Prediction
29000	1450000	Correct Prediction
30000	1500000	Correct Prediction
0	0	Incorrect Prediction
1000	100000	Incorrect Prediction
2000	200000	Incorrect Prediction
3000	300000	Incorrect Prediction
4000	400000	Incorrect Prediction
5000	500000	Incorrect Prediction
6000	600000	Incorrect Prediction
7000	700000	Incorrect Prediction
8000	800000	Incorrect Prediction
9000	900000	Incorrect Prediction
10000	1000000	Incorrect Prediction
11000	1100000	Incorrect Prediction
12000	1200000	Incorrect Prediction
13000	1300000	Incorrect Prediction
14000	1400000	Incorrect Prediction
15000	1500000	Incorrect Prediction
16000	1600000	Incorrect Prediction
17000	1700000	Incorrect Prediction
18000	1800000	Incorrect Prediction
19000	1900000	Incorrect Prediction
20000	2000000	Incorrect Prediction
21000	2100000	Incorrect Prediction
22000	2200000	Incorrect Prediction
23000	2300000	Incorrect Prediction
24000	2400000	Incorrect Prediction
25000	2500000	Incorrect Prediction
26000	2600000	Incorrect Prediction
27000	2700000	Incorrect Prediction
28000	2800000	Incorrect Prediction
29000	2900000	Incorrect Prediction
30000	3000000	Incorrect Prediction

## 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.

A screenshot of a Jupyter Notebook code cell. The code defines a KNeighborsRegressor, creates a parameter grid for n\_neighbors from 1 to 31, uses GridSearchCV with cv=5 and scoring='neg\_mean\_squared\_error' to find the best k. The output shows the best k is 2 with a score of -8269215.208594243.

```
# 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.

✓  
0s

```
# 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

✓  
0s

```
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()
```

```
# 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.