Aisha Mahmoud and Ali Mohammed

Predicting Future of COVID-19 Cases

EECE 481 - Machine Learning

Final Report

Table of Contents:

Table of Contents:	2
Table of Figures:	3
Introduction:	4
Data Explanation:	5
Notes	8
Data Preprocessing	9
Previous Works	13
Data Analysis	15
Methods	17
Results	19
Conclusion:	30
References:	31-33

Table of Figures:

Figure 1 5-7
Figure 2: 13
Figure 3: 15
Figure 4 16
Figure 5 17
Figure 6 19
Figure 7 20
Figure 8 22
Figure 9 23
Figure 11 24
Figure 12: 25
Figure 13 26
Figure 14: 26
Figure 15 26
Figure 16 27
Figure 17: 28
Figure 18 28
Figure 19: 29

Predicting Future of COVID-19 Cases

Introduction:

There is a lot of data out there in the world. Much of it has had time to be deeply studied, and much of it has not. One particular set of information that is relatively new, with new numbers still coming out every day, is the data on the most recent pandemic in human society: The COVID-19 pandemic. Since the first outbreak of COVID-19, over 638 million people around the world have been diagnosed with the illness, and there have been over 6 million deaths [1]. Therefore, many worldwide agencies have been collecting data on the disease throughout its lifespan.

The first cases started in December 2019 and although it appears to have slowed down, cases have still been ongoing ever since [1]. At the beginning of the pandemic, scientists were scrambling to learn more about the disease, especially since it was so new. Today, many years of data exist, and while this is not nearly enough in the large scheme of things, it gives researchers a base to start off with.

For our project, we have decided to split it into two phases. Phase-1 was our initial attempt at creating a linear regression model to predict the total cases using specific features. Since this was our first attempt, we faced many issues with the model that we will explore in the paper. Moreover, we decided to include it to show our understanding and development while going into phase-2. Phase-2 objective was to shift from predicting total cases and use a subset of the data engineered to predict the number of new cases per 100k. Phase-2 included two regression models. We used linear regression and random forest regression. Both phases will be explained thoroughly in the paper.

Data Explanation:

This is the dataset maintained by Our World in Data. Different metrics are provided and described. This dataset has been updated daily throughout the entire COVID-19 pandemic. The start date recorded on the data was 2020-01-23, and the last date on this dataset is 2022-07-04. [2]. Below are the list of variables that will be included in Phase-1 of this project and then engineered for Phase-2.

Variable	Description
Confirmed Cases	
total_cases	Total confirmed cases of COVID-19. Counts can include probable cases, where reported.
new_cases	New confirmed cases of COVID-19. Counts can include probable cases, where reported. In rare cases where our source reports a negative daily change due to a data correction, we set this metric to NA.
new_cases_smoothed	New confirmed cases of COVID-19 (7-day smoothed). Counts can include probable cases, where reported.
total_cases_per_million	Total confirmed cases of COVID-19 per 1,000,000 people. Counts can include probable cases, where reported.
new_cases_per_million	New confirmed cases of COVID-19 per 1,000,000 people. Counts can include probable cases, where reported.
new_cases_smoothed_per_million	New confirmed cases of COVID-19 (7-day smoothed) per 1,000,000 people. Counts can include probable cases, where reported.
Confirmed Deaths	
total_deaths	Total deaths attributed to COVID-19. Counts can include probable deaths, where reported.
new_deaths	New deaths attributed to COVID-19. Counts can include probable deaths, where reported. In rare cases where our source reports a negative daily change due to a data correction, we set this metric to NA.

new_deaths_smoothed	New deaths attributed to COVID-19 (7-day smoothed). Counts can include probable deaths, where reported.
total_deaths_per_million	Total deaths attributed to COVID-19 per 1,000,000 people. Counts can include probable deaths, where reported.
new_deaths_per_million	New deaths attributed to COVID-19 per 1,000,000 people. Counts can include probable deaths, where reported.
new_deaths_smoothed_per_million	New deaths attributed to COVID-19 (7-day smoothed) per 1,000,000 people. Counts can include probable deaths, where reported.
Hospital & ICU	
icu_patients	Number of COVID-19 patients in intensive care units (ICUs) on a given day.
icu_patients_per_million	Number of COVID-19 patients in intensive care units (ICUs) on a given day per 1,000,000 people.
hosp_patients	Number of COVID-19 patients in hospital on a given day.
hosp_patients_per_million	Number of COVID-19 patients in hospital on a given day per 1,000,000 people.
weekly_icu_admissions	Number of COVID-19 patients newly admitted to intensive care units (ICUs) in a given week (reporting date and the preceeding 6 days).
weekly_icu_admissions_per_million	Number of COVID-19 patients newly admitted to intensive care units (ICUs) in a given week per 1,000,000 people (reporting date and the preceeding 6 days).
weekly_hosp_admissions	Number of COVID-19 patients newly admitted to hospitals in a given week (reporting date and the preceeding 6 days).
weekly_hosp_admissions_per_million	Number of COVID-19 patients newly admitted to hospitals in a given week per 1,000,000 people (reporting date and the preceeding 6 days).

Policy Responses	
stringency_index	Government Response Stringency Index: composite measure based on 9 response indicators including school closures, workplace closures, and travel bans, rescaled to a value from 0 to 100 (100 = strictest response).
Reproduction Rate	
reproduction_rate	Real-time estimate of the effective reproduction rate (R) of COVID-19.
Vaccinations	
total_vaccinations	Total number of COVID-19 vaccination doses administered.
people_vaccinated	Total number of people who received at least one vaccine dose.
people_fully_vaccinated	Total number of people who received all doses prescribed by the initial vaccination protocol.
total_boosters	Total number of COVID-19 vaccination booster doses administered (doses administered beyond the number prescribed by the vaccination protocol).
new_vaccinations	New COVID-19 vaccination doses administered (only calculated for consecutive days).
new_vaccinations_smoothed	New COVID-19 vaccination doses administered (7-day smoothed). For countries that don't report vaccination data on a daily basis, we (OWID) assume that vaccination changed equally on a daily basis over any periods in which no data was reported. This produces a complete series of daily figures, which is then averaged over a rolling 7-day window.
total_vaccinations_per_hundred	Total number of COVID-19 vaccination doses administered per 100 people in the total population.

people_vaccinated_per_hundred	Total number of people who received at least one vaccine dose per 100 people in the total population.
people_fully_vaccinated_per_hundred	Total number of people who received all doses prescribed by the initial vaccination protocol per 100 people in the total population.
total_boosters_per_hundred	Total number of COVID-19 vaccination booster doses administered per 100 people in the total population.
new_vaccinations_smoothed_per_million	New COVID-19 vaccination doses administered (7-day smoothed) per 1,000,000 people in the total population.
new_people_vaccinated_smoothed	Daily number of people receiving their first vaccine dose (7-day smoothed).
new_people_vaccinated_smoothed_per_hundred	Daily number of people receiving their first vaccine dose (7-day smoothed) per 100 people in the total population.
Others	
location	Geographical location.
date	Date of observation.
population	Population (latest available values).
population_density	Number of people divided by land area, measured in square kilometers, most recent year available.
median_age	Median age of the population, UN projection for 2020.
aged_65_older	Share of the population that is 65 years and older, most recent year available.
gdp_per_capita	Gross domestic product at purchasing power parity (constant 2011 international dollars), most recent year available.
extreme_poverty	Share of the population living in extreme poverty, most recent year available since 2010.
cardiovasc_death_rate	Death rate from cardiovascular disease in 2017 (annual number of deaths per 100,000 people).
diabetes_prevalence	Diabetes prevalence (% of population aged 20 to 79) in 2017.
female_smokers	Share of women who smoke, most recent year available.
male_smokers	Share of men who smoke, most recent year available.
handwashing_facilities	Share of the population with basic handwashing facilities on premises, most recent year available.
hospital_beds_per_thousand	Hospital beds per 1,000 people, most recent year available since 2010.
life_expectancy	Life expectancy at birth in 2019.
human_development_index	A composite index measuring average achievement in three basic dimensions of human development—a long and healthy life, knowledge and a decent standard of living. Values for 2019, imported from hdr.undp.org.

Figure 1: entire list of variables of data set

All excess mortality variables are removed from the dataset. These variables described the difference between projected and reported number of deaths. Moreover, variables that describe tests were also removed. The data removed will be removed for both phases of

predictions. More information will be followed up on these decisions in the data preprocessing section.

Notes

These notes are provided by the OWID team.

- Confirmed cases and deaths are collected by Johns Hopkins University by date of report, rather than date of test/death. Therefore the number they report on a given day does not necessarily represent the actual number on that date, because of the long reporting chain that exists between a new case/death and its inclusion in statistics. This also means that time series can show sudden changes (negative or positive) when a country corrects historical data, because it had previously under- or overestimated the number of cases/deaths.
- In rare cases where our source for confirmed cases & deaths reports a negative daily change due to a data correction, we set the corresponding metric (new_cases or new_deaths) to NA. This also means that rolling metrics (7-day rolling average, weekly rolling sum, biweekly rolling sum) are set to NA until this missing value leaves the rolling window.
- Due to varying protocols and challenges in the attribution of the cause of death, the number of confirmed deaths may not accurately represent the true number of deaths caused by COVID-19.
- The population estimates we use to calculate per-capita metrics are based on the last revision of the United Nations World Population Prospects. The exact values can be viewed here. In a few cases, we use other sources (see column source in the population

file) when the figures provided by the UN differ substantially from reliable and more recent national estimates. Population estimates for a few subnational locations are taken from national reports, and are stored here.

Data Preprocessing

Phase 1 - Data Preprocessing

In phase 1 of the project, the data preprocessing was limited. We selected 21 features:

- Reproduction_rate
- Population_density
- Median_age
- Aged_65_older
- Gdp_per_capita
- Life_expectancy
- Human_development_index
- Hospital_beds_per_thousand
- Stringency_index
- Aged_70_older
- Male_smokers
- Female_smokers
- Diabetes_prevalence
- Hospital_beds_per_thousand
- Total_tests
- New_tests
- People_vaccinated_per_hundred
- Total_boosters_per_hundred
- Population
- Total_vaccinations
- human_development_index

We only selected these features because of the high null values in the other recorded features (60%-80%). In the selected features, we checked for the null values, but ultimately decided to fill

new_tests with 0's. For the rest of the null values, we decided to take the mean of each feature. After consulting with Dr. Nejib, we came to the conclusion that this technique for processing the data was flawed and further measures needed to be taken to create a better model. Nevertheless, we continued to predict the total cases based on these features with errors and decided to start with an engineered data set for phase-2.

Phase 2 - Data Preprocessing

The OWID COVID-19 dataset contained around 198000 records with 67 columns. These datasets were recorded from the start date of 2020-01-23 and a last date of 2022-07-04. The dataset contained 244 different locations. We swapped the date column into the number of COVID days that have passed for a better point of view on the analysis. The references in the data preprocessing support our decision to drop or fill a specific feature.

We started preprocessing by dropping all countries with no population recorded. Next, we dropped all testing data features as the reporting of each country was not structured and varied. We dropped ICU units and admissions because of the high null rate of 85% [14]. We dropped iso_code and replaced it with row_id. We dropped continent and aged over 70 because the information was not of value to us. We also dropped features including deaths as we were not predicting deaths that happened but only the new cases. We also dropped the missing values from total cases and that reduced the null values in new cases. We also dropped values that are normalized per million. We also dropped male and female smokers because of the high null values of over 60% [14]. Finally, we decided to drop all smoothed values as we did not want to experiment with them in this project.

As for missing values, we handled them by filling new cases nulls with 0's because of the low null rate of 4% [14]. This data was of importance to us because this is the data we are predicting. For vaccinations, we know that they had a huge role in containing the vaccine and therefore a huge role in predicting the number of new cases. Therefore, even though we had a high null value for total vaccinations we decided not to drop it but fill the null values with zeros [14].

Normalization of data

Data for total cases, total vaccinations, total deaths, and people fully vaccinated have been normalized to show the data within 100k of the population. We also select only countries that have a population of over 1 million as many countries with a smaller population had bad reporting of data. This leaves us with a cleaned data frame that's extracted from the engineered one and contains 25 columns with 99856 records.

The columns are as follows:

- location
- covid_days
- total_cases
- new_cases
- reproduction_rate
- total_vaccinations
- people_fully_vaccinated
- stringency_index
- population
- population_density
- median_age
- aged_65_older
- gdp_per_capita
- cardiovasc_death_rate
- diabetes_prevalence
- life_expectancy
- human_development_index
- daily_vaccinations
- daily_people_fully_vaccinated
- total_cases_per_100K

- new_cases_per_100K
- total_vaccinations_per_100K
- people_fully_vaccinated_per_100K
- daily_vaccinations_per_100K
- daily_people_fully_vaccinated_per_100K

There are 177 locations in the cleaned data frame. To proceed with the predictions we will need to drop columns:

- Location
- Total_cases
- New_cases
- Total_vaccinations
- Daily_vaccinations
- People_fully_vaccinated
- Daily_people_fully_vaccinated
- population

We are left with 17 features to create our regression models on.

Multi-collinearity

In regression, multicollinearity is when a predictor is correlated to another predictor. Sometimes the features are related to each other and not just the predictor value. These features become a bit redundant. [15]

We decided to include a collinearity heatmap to show the correlation of the features. We will not perform a principal component analysis in this phase; however, moving forward and getting better results for this model would require us to do so.

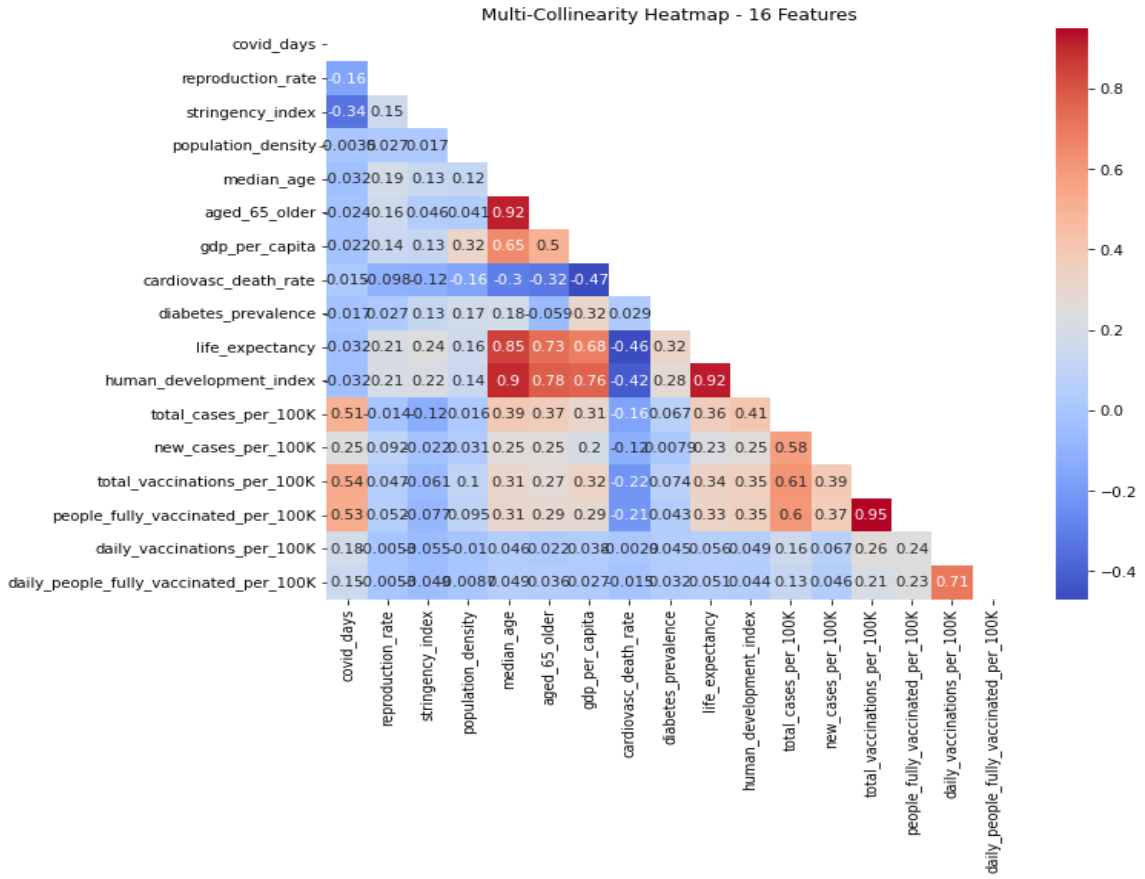


Figure 2: Multi-collinearity heat map

Previous Work

Other regression analyses have been done on COVID data sets. In May 2020, F. Rustam et al published a paper in which they used three different supervised machine learning models for future forecasting of COVID-19. They concluded that linear regression is a sufficient tool for predicting death rates and confirmed cases. They also recommended exponential smoothing for such tests, and concluded that the support vector machine did not perform well with such forecasting. [3] This helped guide us in our decision to use linear regression in our project. Other research done like the one done by Abrar Almalki et. al. focused on multivariable linear

regression to explore the impact of COVID-19 infections and deaths' relationship with health and food access. [4]. This preliminary study done in June 2020 found that there was no correlation between COVID-19 and health issues such as obesity, high cholesterol, and high blood pressure, which is interesting because later studies done in 2021 show that high BMI and cholesterol do in fact lead to a high-risk factor for COVID-19 [4]. This change is to be expected, as in June 2020, when the first study came out, was only a few months into the pandemic when there were a lot of unknowns, whereas the second study had more data and more knowledge to work with.

Multiple datasets have been analyzed, and we selected a COVID-19 dataset from "Our World in Data", a scientific publication. Within this data, we found a large dataset that is constantly updated with a wide set of attributes that are related to countries and people who were affected by the pandemic. It includes location, date, total cases, total deaths, ICU admissions, tests, and many other features. For the purpose of this project, we will focus on location, the number of cases, and the total number of deaths. In this project, we will analyze and use the dataset to predict, given a certain future date, the number of COVID-19 cases there will be present.

Data Analysis

For phase 1, we focused on the features "iso_code", "continent", "location", "date", "total_cases", "new_cases", "total_deaths", and "new_deaths". The dataset required some cleaning, such as dropping null values and converting the date from a string to a date/time format.

	iso_code	continent	location	date	total_cases	new_cases	total_deaths	new_deaths
28	AFG	Asia	Afghanistan	2020-03-23	40.0	6.0	1.0	1.0
29	AFG	Asia	Afghanistan	2020-03-24	42.0	2.0	1.0	0.0
30	AFG	Asia	Afghanistan	2020-03-25	74.0	32.0	1.0	0.0
31	AFG	Asia	Afghanistan	2020-03-26	80.0	6.0	2.0	1.0
32	AFG	Asia	Afghanistan	2020-03-27	91.0	11.0	2.0	0.0
...	...	...	...	...	...	...	...	...
237470	ZWE	Africa	Zimbabwe	2022-11-18	257893.0	0.0	5606.0	0.0
237471	ZWE	Africa	Zimbabwe	2022-11-19	257893.0	0.0	5606.0	0.0
237472	ZWE	Africa	Zimbabwe	2022-11-20	257893.0	0.0	5606.0	0.0
237473	ZWE	Africa	Zimbabwe	2022-11-21	257893.0	0.0	5606.0	0.0
237474	ZWE	Africa	Zimbabwe	2022-11-22	257893.0	0.0	5606.0	0.0

190740 rows × 8 columns

Figure 3 - Dataset table after cleaning

Sequentially, some exploratory data analysis was performed, as a way to learn more about the distribution of data and what information it contained within. The data frame itself had 214 locations in the data frame. The mortality rate of COVID-19 cases in all these locations overall was 2.1%, while the mortality rate in just the UAE was 0.2%. When compared to European countries' mortality rate, the UAE's rate is astoundingly low. This is due to a number of factors, such as the high wealth of these countries, enabling them to better prepare for the pandemic and weather its effects. Wealthy counties can adapt towards higher demand of hospital beds, medicine, sanitary products, and COVID-19 tests much easier. [5]. This isn't the only factor. The gulf region of the Middle East, in which the UAE is located, has about 2.25% of residents above the age of 65, whereas in the European Union, 19.2% of citizens are older than 65. [5]. This is significant as COVID-19 has a much worse effect on elderly people and those with health problems overall. The GCC having a higher young population is another point that

explains why it, and the UAE specifically, had low mortality rates. [5]

```
labels = ['Cases','Deaths',]
sizes = [latest_cases_df['total_cases'].iloc[-1],latest_cases_df['total_deaths'].iloc[-1]]
colors = ['r','y']
explode = [0,0]
plt.figure(figsize=(8,5))
plt.pie(sizes,autopct='%1.1f%%',pctdistance=1.2,explode=explode,colors = colors,startangle=90)
plt.axis('equal')
plt.title('Distribution total cases and total death cases\n\n')
plt.legend(labels=labels,loc='upper right')
plt.show()
```

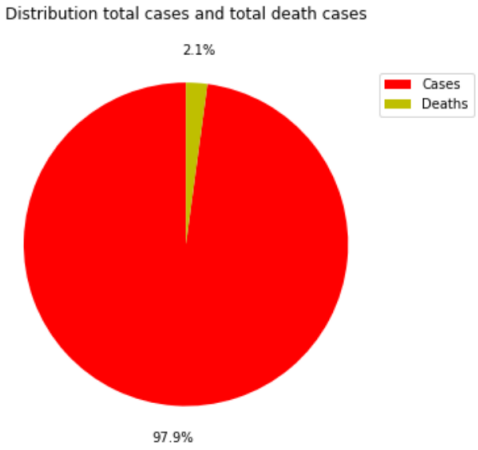


Figure 4: code and plot of mortality rate in the whole world

```
labels = ['Cases','Deaths',]
sizes = [uae_latest['total_cases'].iloc[-1],uae_latest['total_deaths'].iloc[-1]]
colors = ['r','y']
explode = [0,0]
plt.figure(figsize=(8,5))
plt.pie(sizes,autopct='%1.1f%%',pctdistance=1.2,explode=explode,colors = colors,startangle=90)
plt.axis('equal')
plt.title('Distribution total cases and total deaths in the UAE\n\n')
plt.legend(labels=labels,loc='upper right')
plt.show()
```

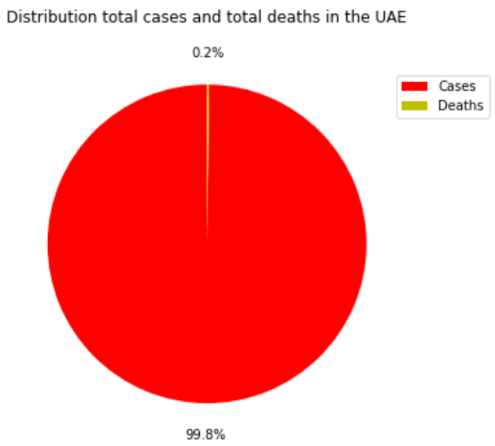


Figure 5: Mortality rates in UAE

Methods

Linear regression is an area of machine learning that is used for "finding the relationship between the two variables and predicting future results based on past relationships"[6]. There is simple linear regression which focuses on just one independent variable and one dependent variable, and there is multi-variable linear regression that tests multiple variables at once. [6] This area of machine learning is a supervised algorithm, meaning it uses data that has been labeled. It is useful for seeing what variables are linked to each other, however, it isn't the best at

explaining complicated and non-linear datasets [6]. It works by finding the line that best fits the data by mapping numeric inputs to numeric outputs [7]. The cost of this function, also known as the error, is how far away the true data points are from the regressor line drawn. Other metrics can be used but Euclidean distance is the most used often, and it takes the distance between the predicted x and actual x, squares it, then divides by the total number of data points[7].

A random forest regression is an algorithm that uses ensemble learning for its predictions. Ensemble learning is when the model takes information from many different classifications or regressions done and becomes an average of that, making it more accurate. It can be thought of as an ensemble of multiple decision trees, hence the term “forest”. Each decision tree makes its own prediction, and the algorithm takes the average of all of these decision tree predictions. The average is the final result[8]. Because it uses information from many different decision trees at once rather than from just one tree or one regressor, it highly improves accuracy. As with any algorithm, it has its advantages and disadvantages. “Between linear regression and random forests, “Generally, Random Forests produce better results, work well on large datasets, and are able to work with missing data by creating estimates for them”[8]. It works well for outliers and for large data sets[9]. However, one disadvantage it has is its inability to perform extrapolation[8]. This means that it cannot predict any values outside of the training set maximum and minimum values. This makes sense as it uses decision trees to predict, which are basically just numerous “if-else” statements. It classifies what it already knows. It cannot go outside of these bounds of the information given to it in if-else statements. To deal with this problem, linear models and deep learning models are recommended to supplement the random forest algorithm [8]. It is used in Python code with sklearn.ensemble.RandomForestRegressor() [10].

Results

Results for Phase 1:

For phase 1, to create the regressor, the data was split into X and y for the total cases in the UAE. train_test_split was imported from the package sklearn.model_selection to split the data. We used a test size of 0.3, meaning that 70% of the data is used for training, and 30% of the data will be used for testing. Linear regression can be measured using the R-squared, or coefficient of determination. This statistical measure explains how well the regression fits the data. It is literally variance explained by the model divided by the total variance [11]. The lower the R-squared score, the less of the data it explains, so a higher score means the regressor fits the data more. It is always a number between 0 and 1.

After training the data then testing it, the regressor's score was 0.9099, which means the regressor accurately guessed the test data 90% of the time (as seen in Figure 6).

```
# Create the regressor: reg
reg= LinearRegression()

# Fit the regressor to the training data
reg.fit(X_train,y_train)

# Predict on the test data: y_pred
y_pred=reg.predict(X_test)

#Score the model
reg.score(X_test,y_test)
```

```
print("R^2: {}".format(reg.score(X_test, y_test)))
rmse = np.sqrt(mean_squared_error(y_test,y_pred))
print("Root Mean Squared Error: {}".format(rmse))
```

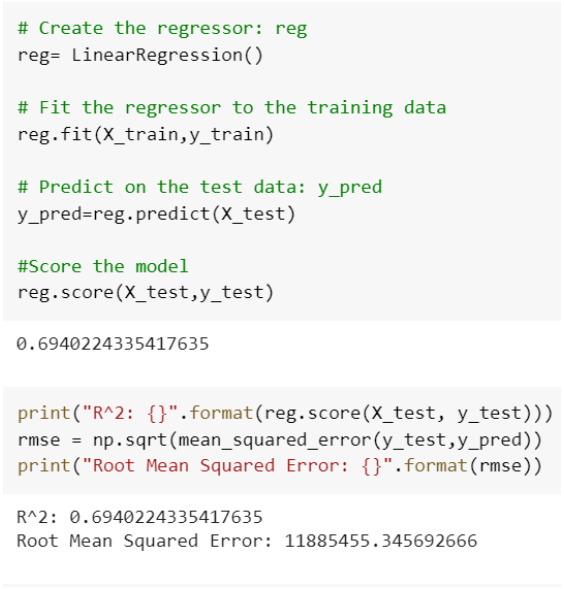
```
R^2: 0.9099028444212713
Root Mean Squared Error: 103956.68763779856
```

Figure 6: Constructing and testing regressor, with RMSE

The R-squared for the regressor for the total cases in the UAE was 0.9000. The root mean square error uses Euclidean distance to see how far prediction points are away from the true measured values [12]. The root mean square error for our regressor was 103956.69. One reason this number might be high is that since it uses the Euclidean distance method for computations, it

is highly sensitive to wide ranges of data, as is the case with distance-based calculations. It works best when the data is scaled first.

This method was repeated by expanding our data set focus from the UAE to the whole world. The dataset was checked for null values. In place of null values, fillna() was used to replace null values with 0, to enable the mean to be calculated. After splitting the data and creating a regressor for it and fitting it to the training data, the test score was 69.40%. The R-squared score for this was 0.34, and the root mean squared error was 11,885,455.34. This root mean squared error being higher than the RMSE for the UAE model regressor is to be expected for the reason mentioned above; Euclidean distance is sensitive to wide ranges of numbers and of course, using data from all over the world over just a small part of the world (UAE) means that the RMSE will be higher.



```
# Create the regressor: reg
reg= LinearRegression()

# Fit the regressor to the training data
reg.fit(X_train,y_train)

# Predict on the test data: y_pred
y_pred=reg.predict(X_test)

#Score the model
reg.score(X_test,y_test)
```

```
0.6940224335417635
```

```
print("R^2: {}".format(reg.score(X_test, y_test)))
rmse = np.sqrt(mean_squared_error(y_test,y_pred))
print("Root Mean Squared Error: {}".format(rmse))
```

```
R^2: 0.6940224335417635
Root Mean Squared Error: 11885455.345692666
```

Figure 7: Constructing, training, and testing linear regressor for world data

Overall, the linear regressions performed on phase 1 of this COVID-19 dataset had a test score in the 90% percentile range for UAE, which is acceptable, and for the rest of the world, it

scored almost 70%. This isn't the worst but there is room for improvement. One way to improve this test number is to re-look at the features included and take out ones that may already be correlated by looking at a scatter plot of them. Another way to improve this score is to scale the data before performing regression on it. Next, we will perform these same regressions with the random forest algorithm instead of a linear regression algorithm to try to improve these numbers.

Results for Phase 2:

Phase 2 includes using the engineered data set mentioned in the data preprocessing section. After performing all the preprocessing we end up with 17 features. This phase's objective was to predict new cases per 100k. We use two regression algorithms; linear regression and random forest. We then created a model for the 150 countries in our data frame and another model just for the UAE.

Linear regression results

World model linear regression

We used train test split with the test size of 25% and train size of 75%. This is the default value of the function. Moreover, we had the random state = 0 to be able to replicate the model.

```
y = new_df1.new_cases_per_100K #target
X = new_df1.drop(["new_cases_per_100K"], axis=1) #features
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
```

Figure 8: setting up X and y to train and test

We then proceed to create the linear regression model by feeding X_train and y_train. We can obtain y_pred by testing the model on the testing set.

```
In [92]: # Create the regressor
         reg= LinearRegression()

         # Fit the regressor
         reg.fit(X_train,y_train)

         # Prediction on test
         y_pred=reg.predict(X_test)
         y_pred

Out[92]: array([ -3.74328685,  19.52472322, -10.41419329, ...,  -1.8397376 ,
                 -1.29768774,  13.62841595])
```

Figure 9: creating regressor

We can find how this model performed by getting the R-squared score while also getting the Root Mean Squared error.

```
In [90]: #Score of the regressor
         reg.score(X_test,y_test)

Out[90]: 0.34535126082596923
```

```
In [51]: print("R^2: {}".format(reg.score(X_test, y_test)))
         rmse = np.sqrt(mean_squared_error(y_test,y_pred))
         print("Root Mean Squared Error: {}".format(rmse))

         R^2: 0.34535126082596923
         Root Mean Squared Error: 41.56905716608328
```

Figure 10: Code for RMSE

The score of the model was 0.34 while the root mean squared error was 41.6. These were the results for the dataframe that contained 150 countries. Now, we will create a different model that includes only the UAE.

UAE Random Linear Regression model

We used the same train and test size, and we can see that the model score using only data from the UAE which included 680 records starting from covid day 78 up to 817. We can see that the UAE model performed much better than the world model.

```
In [62]: X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)

In [63]: # Create the regressor
         reg= LinearRegression()

         # Fit the regressor to training data
         reg.fit(X_train,y_train)

         # Predict on the test data
         y_pred=reg.predict(X_test)

         #Score
         reg.score(X_test,y_test)

Out[63]: 0.9999999999904423

In [64]: print("R^2: {}".format(reg.score(X_test, y_test)))
         rmse = np.sqrt(mean_squared_error(y_test,y_pred))
         print("Root Mean Squared Error: {}".format(rmse))

         R^2: 0.9999999999904423
         Root Mean Squared Error: 2.8368004372094175e-05
```

Figure 11: UAE's regression score and RMSE

The score of the model was 0.99 while the RMSE was 2.8.

Random Forest Regressor Results

Now, we will proceed with the second regression algorithm that we used, which is Random Forest Regressor. Random forest uses a number of decision trees on various samples of the data and averages them to get the output [8].

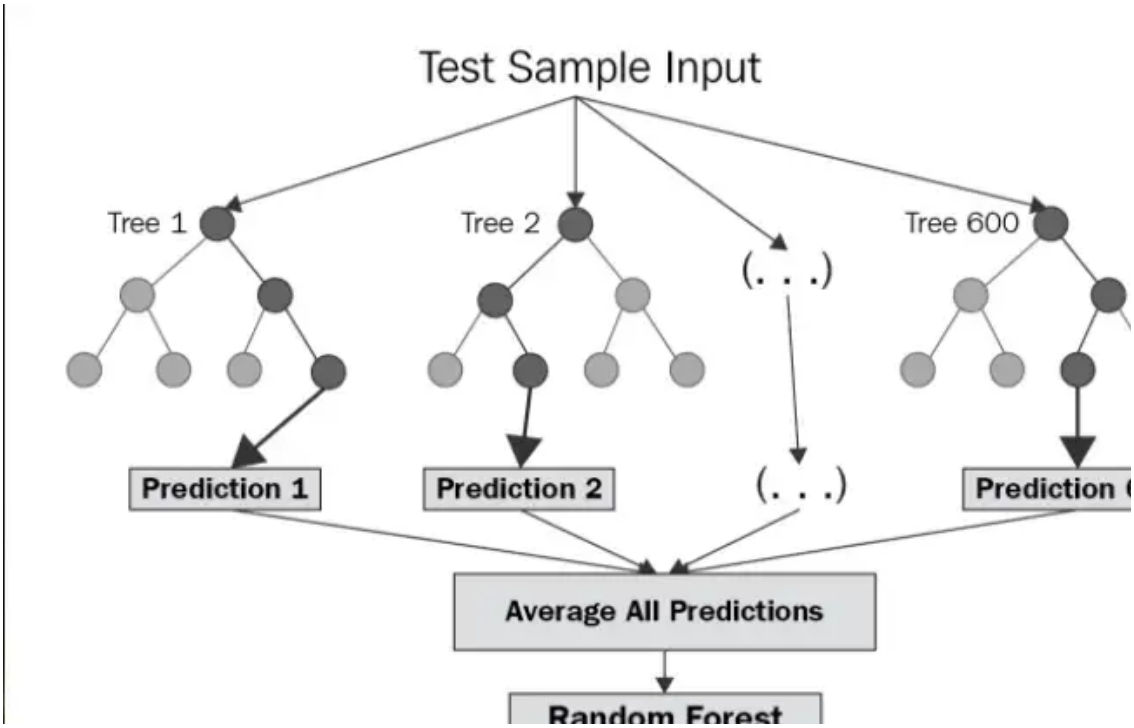


Figure 12: Random forest

World Random Forest Regression model

For this model, we used the default number of trees (ten) because every time we tried to increase the number of trees. The notebook would freeze up. After lots of searching online about this issue, we found that the model requires a lot of computation power and the notebook could not handle it with a large number of trees because of the high level of computation needed to create the model. Nevertheless, we proceeded to feed the model our X_train and y_train using train test split with the default setting.

```
# Create a RF regressor
rf_model = RandomForestRegressor(random_state=0)

# fitting the model
rf_model = rf_model.fit(X, y)
```

Figure 13: creating random forest regressor

We tested our model to give a prediction y_pred on the testing set.

```
# predict RFmodel on test data
y_pred = rf_model.predict(X_test)
y_pred
```

```
array([ 0.290251, 19.425899,  0.423156, ...,  4.125787,  0.019885,
        0.320871])
```

Figure 14: y_pred array

We got the score of the model on the training set and the testing set.

```
# training score for rf_model
training_score = rf_model.score(X_train, y_train)
print(f"The training score for the Random Forest Regression model on 16 features is {training_score: .2
```

```
The training score for the Random Forest Regression model on 16 features is  0.97.
```

```
 #test score for rf_model
testing_score = rf_model.score(X_test, y_test)
print(f"The testing score for the Random Forest Regression model on 16 features is {testing_score: .2f}
```

```
The testing score for the Random Forest Regression model on 16 features is  0.97.
```

Figure 15: testing random forest model

Training score and testing score were equal to 0.97.

We visualized the prediction of the model and used a scatter plot to plot the results.

```
# Visualize the predication against the actual values
plt.figure(figsize=(10, 8))
plt.scatter(X_test['covid_days'].values, y_test, facecolors='none', edgecolors='r')
plt.scatter(X_test['covid_days'].values, y_pred, facecolors='none', edgecolors='blue')

plt.title('Random Forest Regression on 16 Features')
plt.xlabel('COVID Days')
plt.ylabel('New Cases /100,000')
plt.show()
```

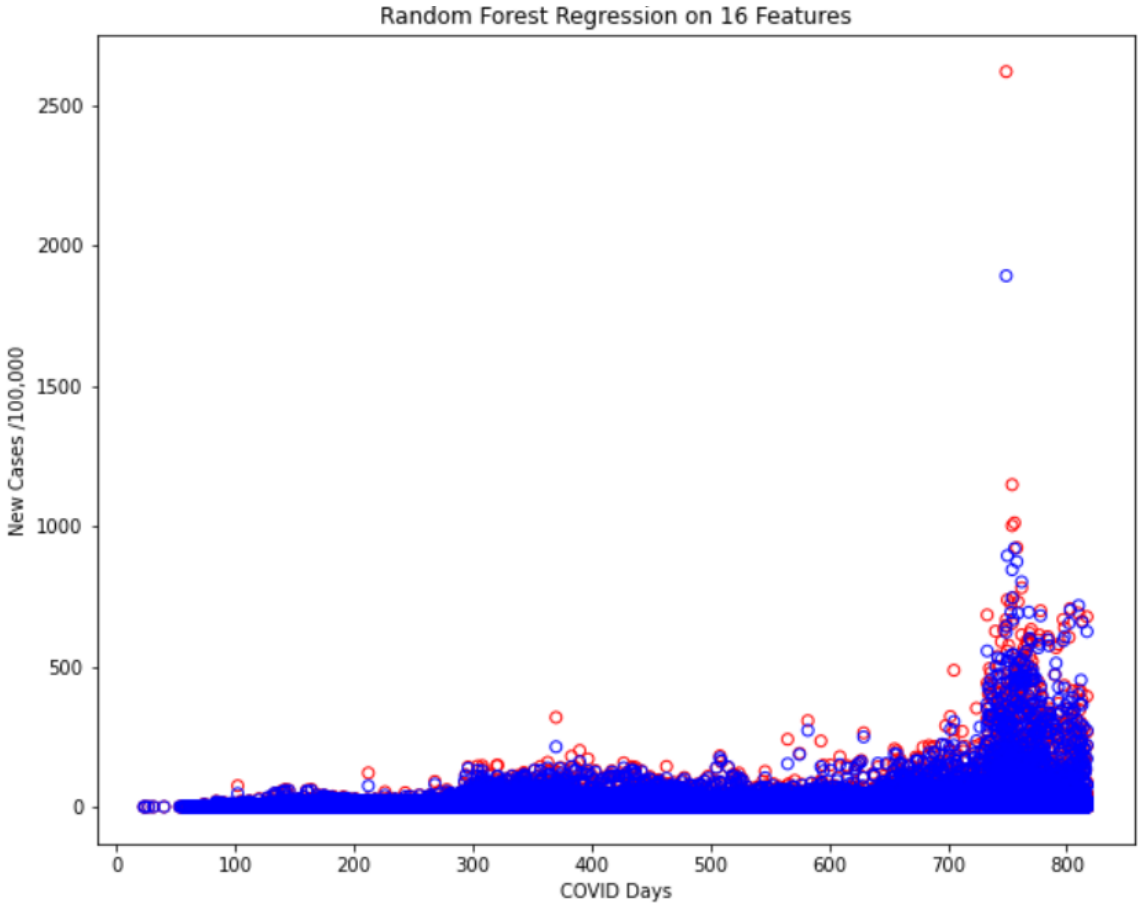


Figure 16: code and scatterplot for random forest regression

Our model predicted the new number of cases per 100k with an accuracy of 97% and Root Mean Squared Error at 9.024. These results were for the world model of 150 countries.

Now we will build a separate random forest regressor for the UAE.

UAE Random Forest Regression model

```
# Create a Random Forest Regression model for uae
rf_model_uae = RandomForestRegressor(random_state=0)

# Training the model
rf_model_uae = rf_model.fit(X, y)
```

```
# evaluate on test
y_pred = rf_model_uae.predict(X_test)
y_pred
```

```
array([ 4.722299,  0.685799, 35.34672 ,  3.9051  , 39.555583, 14.932717,
        6.002046, 15.494401, 30.197957,  0.869879,  2.114106, 22.499879,
        4.78159 ,  6.215928,  0.899598, 25.810794,  0.591711, 11.73499 ,
```

Figure 17: UAE random forest regression

The model R-squared score and root mean squared error are shown as follows:

```
r2 = r2_score(y_test, y_pred)
RMSE = metrics.mean_squared_error(y_test, y_pred, squared=False)

print(f"R^2: {r2: .3f}")
print(f"Root Mean Squared Error: {RMSE: .3f}")
```

```
R^2:  1.000
Root Mean Squared Error:  0.026
```

Figure 18: UAE R-squared and RMSE

We can see that the model had a perfect R-squared score of 1. Now that doesn't make much sense especially because the model score was on the tested set. We would need more time to decipher what has happened to fully understand why such an R-squared score would be outputted.

We visualize the data:

```
In [107]: # Visualize the predication against the actual values
          plt.figure(figsize=(10, 8))
          plt.scatter(X_test['covid_days'].values, y_test, facecolors='red', edgecolors='red')
          plt.scatter(X_test['covid_days'].values, y_pred, facecolors='blue', edgecolors='blue')

          plt.title('Random Forest Regression on 16 Features')
          plt.xlabel('COVID Days')
          plt.ylabel('New Cases /100,000')
          plt.show()
```

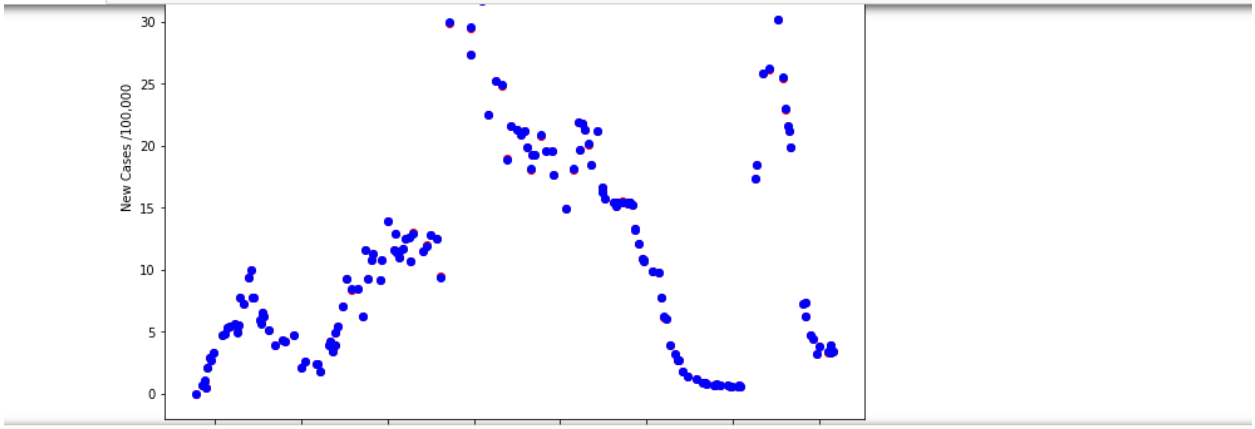


Figure 19: code and scatterplot for UAE random forest

This concludes our results section.

Conclusion:

In this study, both linear regression and random forest algorithms were used to perform regression on a COVID-19 dataset to predict future COVID-19 cases. The significance of this cannot be understated as predicting future cases of any disease is greatly helpful for preparation purposes in healthcare as well as the welfare of everyday citizens. Through our research, we found that the Random Forest algorithm is more accurate at forecasting the future than simple linear regression. We highlighted the importance of data preprocessing and how it really impacts the accuracy of the regressor, as shown in the difference of scores between phases and between algorithms. Many techniques such as PCA could optimize our results, but due to timing constraints, we could not pursue such techniques to fully optimize the data. However, in completing this study we can infer a lot of information about COVID-19 and how many different factors contribute to predicting or forecasting the future number of cases. This study could be replicated to forecast the number of deaths by using the features dropped. There is a lot that we can do with machine learning and data analysis and it's a topic that we are thrilled to pursue.

References

[1] "Who coronavirus (COVID-19) dashboard," *World Health Organization*. [Online]. Available: https://covid19.who.int/. [Accessed: 02-Dec-2022].

[2] Edouard Mathieu, Hannah Ritchie, Lucas Rodés-Guirao, Cameron Appel, Charlie Giattino, Joe Hasell, Bobbie Macdonald, Saloni Dattani, Diana Beltekian, Esteban Ortiz-Ospina and Max Roser (2020) - "Coronavirus Pandemic (COVID-19)". Published online at OurWorldInData.org. Retrieved from: 'https://ourworldindata.org/coronavirus' [Online Resource] (accessed via GitHub: https://github.com/owid/covid-19-data/blob/master/public/data/owid-covid-data.csv)

[3] F Rustam, A. A. Reshi, A. Mehmood, S. Ullah, B.-W. On, W. Aslam, and G. S. Choi, "Covid-19 future forecasting using supervised machine learning models," *IEEE Access*, vol. 8, pp. 101489–101499, 2020. https://ieeexplore.ieee.org/ielx7/6287639/8948470/09099302.pdf

[4] M. Sarmadi, S. M. Ahmadi-Soleimani, M. Fararouei, and M. Dianatinasab, "Covid-19, body mass index and cholesterol: An ecological study using global data," *BMC Public Health*, vol. 21, no. 1, 2021. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8453032/

[5] A. Rimawi and A. Rimawi, "Covid-19-associated mortality across the countries of the Gulf Cooperation Council and how it compares to Europe: A comparative study," *Qatar Medical Journal*, vol. 2021, no. 2, 2021.

[6] "Linear regression," *CORP-MIDS1 (MDS)*, 05-Aug-2022. [Online]. Available: https://www.mastersindatascience.org/learning/machine-learning-algorithms/linear-regression/. [Accessed: 02-Dec-2022].

[7] A. Al-Masri, “How does linear regression actually work?,” *Medium*, 03-Aug-2021. [Online]. Available: https://towardsdatascience.com/how-does-linear-regression-actually-work-3297021970dd. [Accessed: 02-Dec-2022].

[8] D. Mwiti, “Random Forest regression: When does it fail and why?,” *neptune.ai*, 14-Nov-2022. [Online]. Available: https://neptune.ai/blog/random-forest-regression-when-does-it-fail-and-why. [Accessed: 02-Dec-2022].

[9] ZinHai, Li. “Using “random forest” for classification and regression”. Chinese Journal of Applied Entomology, 2013 Vol,50 No4 pp 1190-1197 ref.23. Institute of Zoology. https://www.cabdirect.org/cabdirect/abstract/20133388344

[10] “Sklearn.ensemble.randomforestregressor,” *scikit*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html. [Accessed: 02-Dec-2022].

[11] “How to interpret R-squared in regression analysis?,” *How to Interpret R-squared in Regression Analysis?* [Online]. Available: https://www.knowledgehut.com/blog/data-science/interpret-r-squared-and-goodness-fit-regression-analysis. [Accessed: 02-Dec-2022].

[12] “Root mean square error (RMSE),” *C3 AI*, 28-Sep-2021. [Online]. Available: https://c3.ai/glossary/data-science/root-mean-square-error-rmse/. [Accessed: 02-Dec-2022].

[13] Edouard Mathieu, Hannah Ritchie, Lucas Rodés-Guirao, Cameron Appel, Charlie Giattino, Joe Hasell, Bobbie Macdonald, Saloni Dattani, Diana Beltekian, Esteban Ortiz-Ospina

and Max Roser (2020) - "Coronavirus Pandemic (COVID-19)". Published online at OurWorldInData.org. Retrieved from: 'https://ourworldindata.org/coronavirus' [Online Resource]

[14] A. Swalin, “How to handle missing data,” Medium, 19-Mar-2018. [Online]. Available: https://towardsdatascience.com/how-to-handle-missing-data-8646b18db0d4. [Accessed: 02-Dec-2022].

[15] J. Frost, “Multicollinearity in regression analysis: Problems, detection, and solutions,” Statistics By Jim, 22-Jul-2022. [Online]. Available: https://statisticsbyjim.com/regression/multicollinearity-in-regression-analysis/. [Accessed: 02-Dec-2022].