STAT390 Data Science Project Midterm Report

Project Group:
NO Work In Main Branch

Team members:
Danling Zhou, Mingyi Gong, Yiqi Zhu

Project Title:
Predicting daily new COVID cases in countries with top 10 GDP per capita

Data Sources:
We used the following datasets from the database through this link:
https://health.google.com/covid-19/open-data/raw-data

Name	Rows	Columns
epidemiology	12525825	10
demographics	21689	19
economy	404	4
emergency_declaration	8364	104
geography	22130	8
health	3504	14
mobility	6321226	8
search_trends	2713929	424
vaccination	2545118	32
government_response	303969	22
world_bank	215	1405
key	22963	15

Combining table and select rows of interest

The tables were joined by common location keys and dates (if applicable). Columns with more than 20% missing values (or more than 10% missing in the world bank dataset) were dropped. Admittedly, dropping columns based on an arbitrary standard is not the optimal approach, but the initial aggregate table was so large that it overwhelmed our computing resources and columns with too many missing columns are likely to cause bias in subsequent data processing and imputation. In retrospect, we think that all the important information, such as vaccination, is retained in our final dataset. We selected the following countries in our final dataset because they ranked top 10 in terms of GDP per capita in 2022: Luxembourg, Ireland, Norway (by region), Switzerland (by canton), Singapore, Qatar, Iceland, Denmark, Australia (by state), United States (by state). We suggest that these countries are similar in terms of economic strength and living standards so that a single set of models may be able to predict the progression of the disease. The combined table has 97,184 rows and 198 columns. The target variable is newly confirmed cases (new_comfirmed) every day in each country/sub-region.

Data preprocessing and missing value imputation

Dropping data

The combined dataset was assessed for missing values. There were 37 columns with missing values. The rows with new_confirmed missing were dropped because that is the target variable.

The following columns were dropped because they contained too much missing data:

lawatlas_mitigation_policy (88920 rows missing), search_trends_infection (43102 rows missing), search_trends_pain (43102 rows missing), SE.PRE.DURS (Preprimary education, duration (years), 54218 rows missing)

The following columns were dropped because of both too much missing data (more than 43000 rows missing) and unexplainable label-encoding presented in the original dataset:

school_closing, workplace_closing, cancel_public_events, restrictions_on_gatherings, public_transport_closing, stay_at_home_requirement, restrictions_on_internal_movement, international_travel_controls, income_support, debt_relief, public_information_campaigns, testing_policy, contact_tracing, facial_coverings, vaccination_policy, stringency_index

cumulative_confirmed was dropped because this is highly correlated with the target variable.

The following columns were dropped because they are unlikely to provide useful information for prediction and/or their information is already included in the location key and country.

place_id, subregion1_code, subregion1_name, subregion2_code

These columns, though not missing, were also dropped because they are also unlikely to provide useful information for prediction and/or their information is already included in the location key and country. For location, there is a column called location_key_x that contains the country and subregion (when applicable) and a country_code column indicating the country.

wikidata_id, location_key_y, iso_3166_1_alpha_2, iso_3166_1_alpha_3, country_name, location_key

Imputing missing values

The following columns were imputed. Considering the time-series nature of the dataset and cumulative nature of some features, three stages of imputation were performed on the following columns.

1. Cumulative data imputation

Columns involved: cumulative_deceased, cumulative_persons_fully_vaccinated

The dataset was grouped by location key and ordered ascendingly by time. The non-missing cumulative data were first checked if the cumulative data was increasing monotonically with the passage of time. When the cumulative data decreased in the dataset, data were matched with the cumulative data from the previous day. The missing values were then forward filled, and the remaining missing data (likely before anyone died from COVID or before anyone was vaccinated) were filled with 0. To ensure that forward filling did not introduce new decreasing cumulative data, the monotonic increasing check function was applied to the imputed dataset again. This turned out to be an issue for the cumulative_persons_fully_vaccinated column but not the cumulative_deceased column.

2. Daily data imputation

Columns involved: new_deceased, new_persons_fully_vaccinated

The ordered dataset was grouped by location key. These columns were re-calculated by subtracting the cumulative data from the previous day from the cumulative data from that day. The first day was then filled with 0.

3. KNN imputation

Columns involved: population_male, population_female, life_expectancy, AG.LND.CROP.ZS (permanent cropland, as percentage of land area), SP.RUR.TOTL.ZG (rural population growth, as annual percentage), mobility_workplaces

The dataset was split into training (date < 2022-01-01, 70829 rows) and testing (date >= 2022-01-01, 26265 rows). The target variable was removed from both datasets to target variable leakage during imputing.

Minmaxsclaler() was used to scale the training and testing datasets. KNNImputer (n_neighbors = 3, limited by the computing resources for imputation) was used to fit and impute the missing values in the training dataset and then impute the missing values in the testing dataset. Both datasets were then inversely transformed using the Minmaxsclaler() to obtain the data at the original scale. The unique values of population_male, population_female, life_expectancy, AG.LND.CROP.ZS and SP.RUR.TOTL.ZG were compared between the training and testing datasets to ensure that they were imputed consistently in the training and testing datasets. These values, by the information provided by the original database, are fixed for a region and not supposed to change with time.

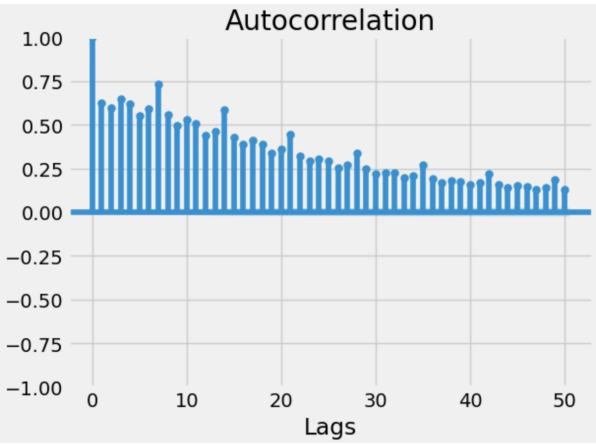
The imputed dataset has 165 columns, excluding the target variable.

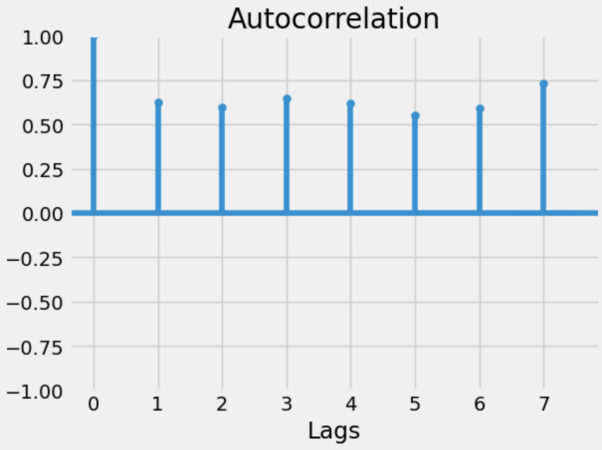
Time-series feature engineering

Training and testing data were merged back together after imputing missing values in preparation for the feature engineering specifically for time-series analysis.

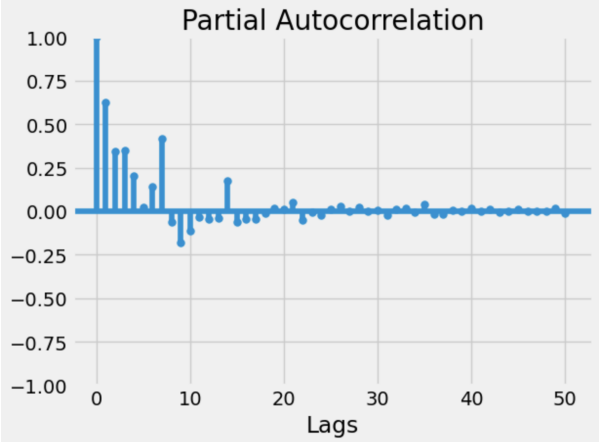
1. Autocorrelation and Partial Correlation

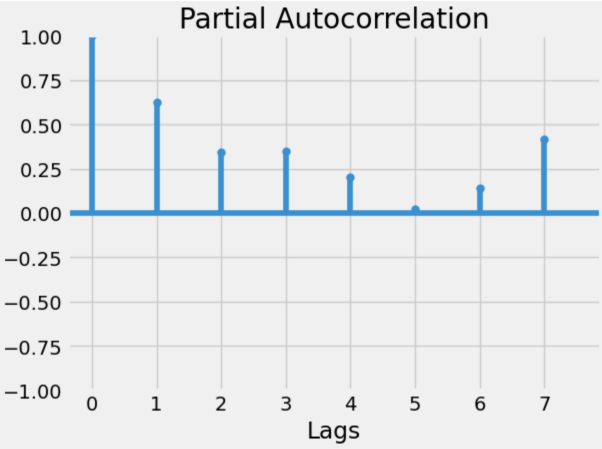
Pearson's correlation coefficient was used to summarize the correlation between variables, assuming the distribution of each variable fits a Gaussian (bell curve) distribution. AutoCorrelation Function was then used to visualize the autocorrelation of the time series by lag.





Based on the graph on the left, a lag of 0-7 was selected for further examination, as shown on the right. Similarly, the partial autocorrelation was visualized below.





Autocorrelation measures the overall linear dependence between a time series and its lagged values, making it useful for detecting any form of serial correlation in the data, while partial autocorrelation provides a more precise measure by isolating the direct relationship between the current observation and its past lags, helping to identify the order of an underlying autoregressive process more accurately in time series analysis. Based on both autocorrelation and partial autocorrelation, the peaks are: lag-1, lag-3, lag-7. This finding guided the later addition of lags features and rolling window statistics.

2. Add Lags Features

According to the plots above, the most relevant lag days are 1, 3, and 7. Therefore, three new features were added to the original dataset - new_confirmed_lag1, new_confirmed_lag3, new_confirmed_lag7 - by shifting the new_confirmed values by 1, 3, 7 days respectively.

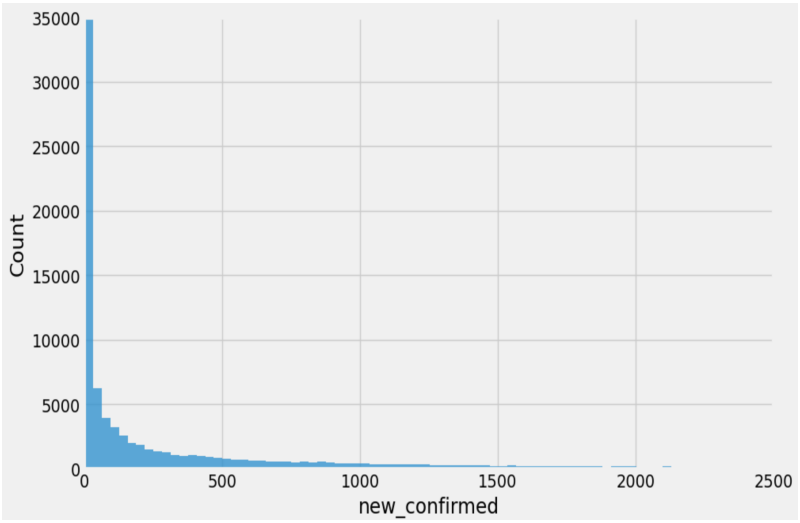
3. Add rolling window statistics

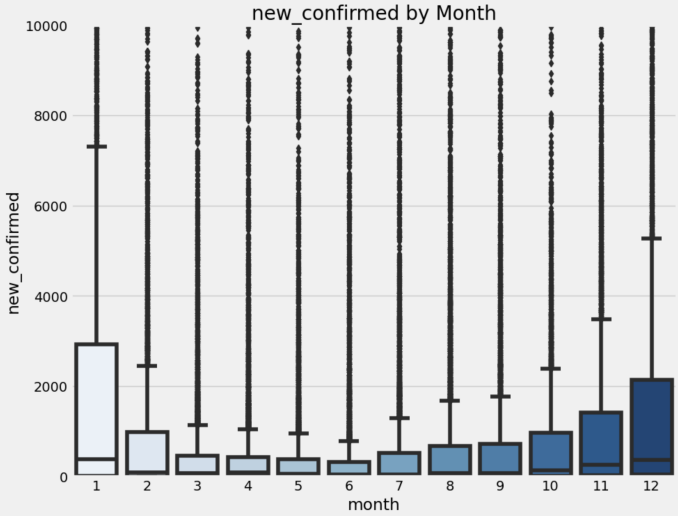
Accordingly, four groups of rolling window statistics were added to the dataset. The windows are 1, 3, 7, and the statistics include mean, standard deviation, minimum and maximum. A total of 12 features were added.

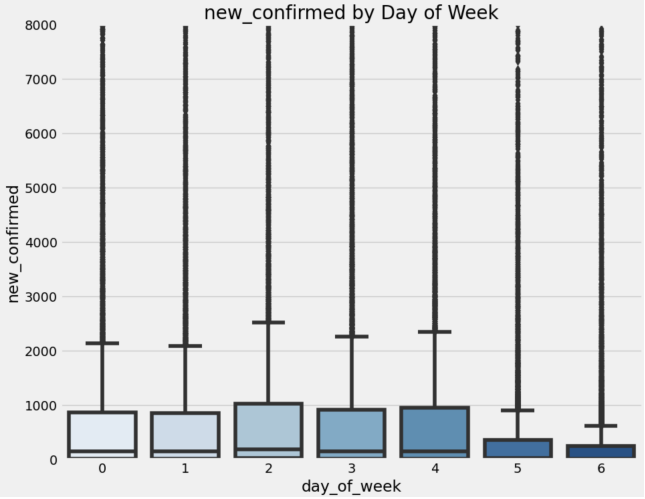
4. Datetime Features

Next, the date column was utilized to generate more datetime related features including day_of_week, day_name, quarter, month, year, day_of_year, day_of_month, week_of_year, season. Importantly, there is one country in our dataset that is in the Southern Hemisphere. Therefore, the seasons are the opposite for that country.

EDA





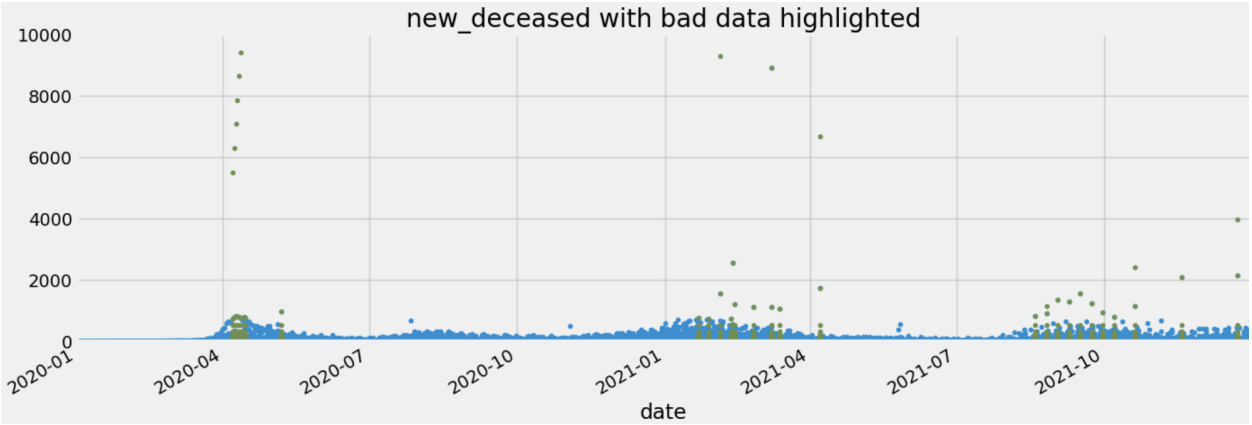


Finding Outliers

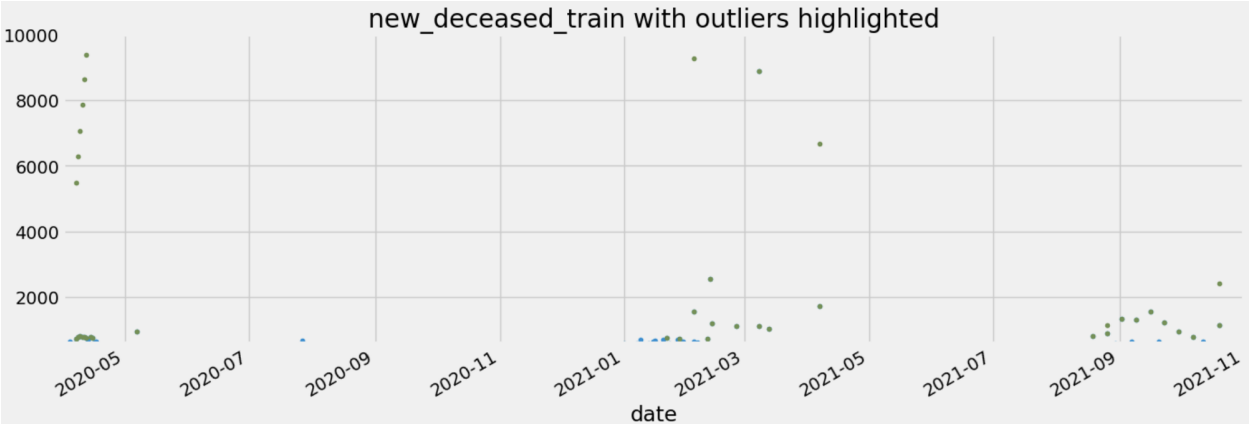
To prepare for ARIMA and LSTM models that will predict based on time series data, we want to remove outliers in our datasets. We first coded with the Kats package that was developed by Facebook Research. However, there were difficulties in pip the Kats package in both online Colab and offline conda environments. Thus, we used the method Professor Shi discussed in the Lecture. In all columns, new_deceased, cumulative_deceased, new_persons_fully_vaccinated, cumulative_persons_fully_vaccinated,mobility_workplaces will change with time. However, when trying to find outliers, cumulative_deceased and cumulative_persons_fully_vaccinated will select all data points after the threshold. Thus, we mainly focus on new_deceased, new_persons_fully_vaccinated and mobility_workplaces ,which we will elaborate in later paragraphs.

We calculated lower and higher outlier bonds for all three features. However, because of the huge amount of zeros in our dataset, the calculated thresholds do not help us identify outliers efficiently. We then chose to tune the threshold manually.

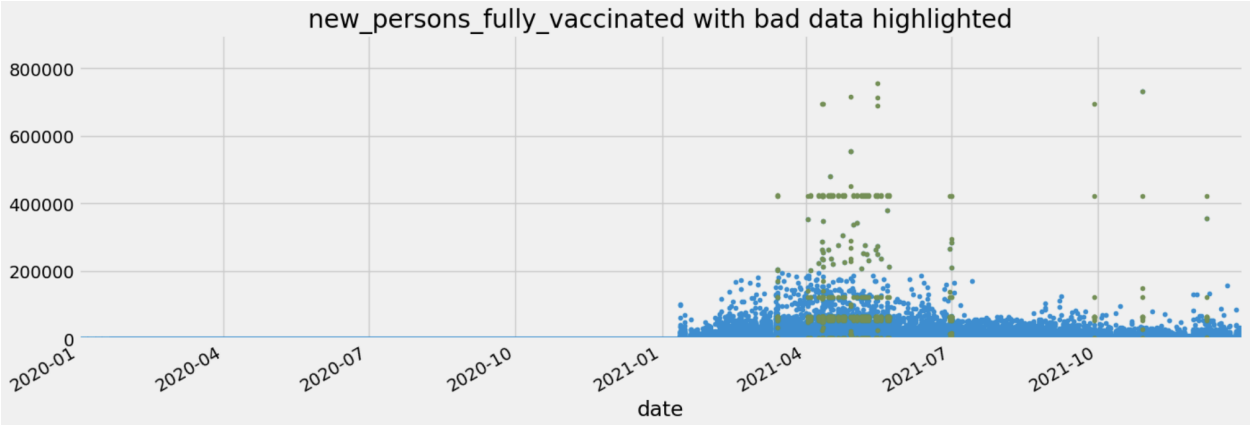
For new_deceased, I sorted the values to learn more about the general trend for this specific feature. I started drawing the graph “new_deceased with bad data highlighted”: all new_deceased data data points are demonstrated in blue points and I highlighted dates with abnormal values with green color. By tuning the y_lim and the threshold for query, I decided the specific values that will best present the outliers in the graph (ylim = (0, 10000); new_deceased > 700).



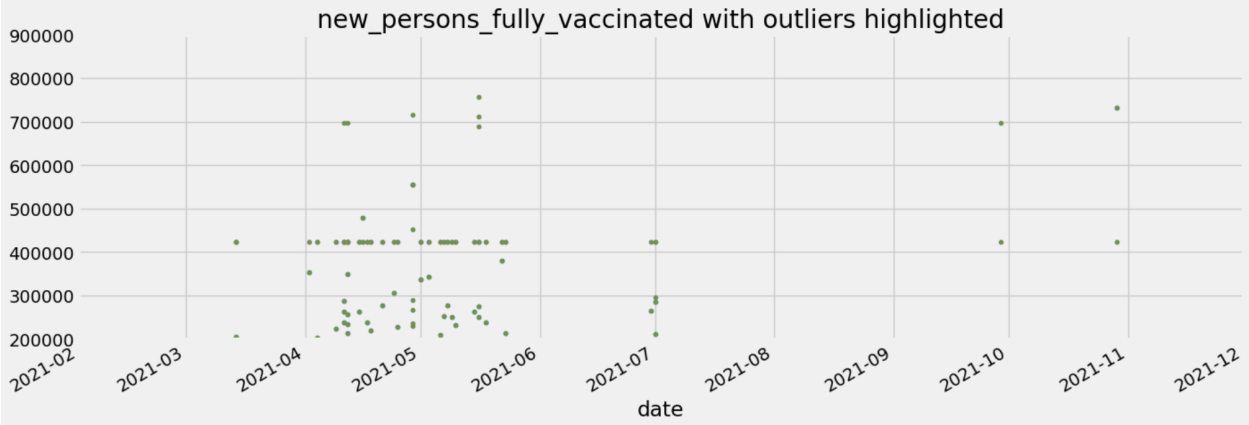
Next, I left all the other values out and drew “new_deceased with outliers highlighted” with only new_deceased > 700. Then I updated the datasets by removing all outliers identified in new_deceased.



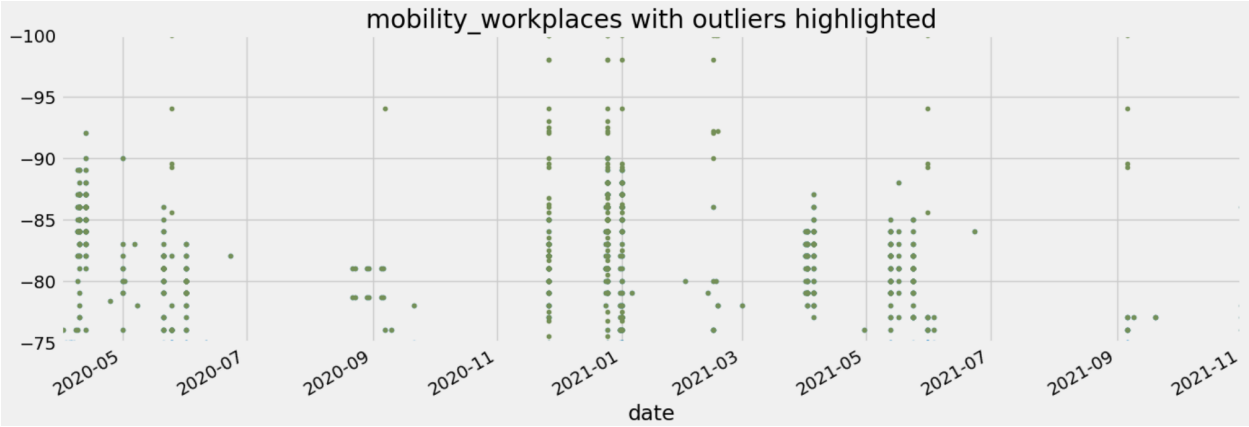
For new_persons_fully_vaccinated, we used the same approach. I first drew the “new_persons_fully_vaccinated with bad data highlighted” and tuned the y_lim as (0, 900000) and the threshold for query as new_persons_fully_vaccinated > 200000.



For this feature, the outliers concentrate near 2021-04-01 to 2021-07-01. The reason could be that there were more vaccines available and many people got the chance to be fully vaccinated. Last but not least, I drew “new_persons_fully_vaccinated with outliers highlighted” and updated our datasets.



For mobility_workplaces, we used the same approach. I first drew the “mobility_workplaces with bad data highlighted” and tuned the y_lim as (-100, 50) and the threshold for query as mobility_workplaces < -75. Last but not least, I drew “mobility_workplaces with outliers highlighted” and updated our datasets.



Challenges:

There are many potential errors in the dataset caused by data collection, including missing data and data that make no sense (like decreasing cumulative data). Despite our effort in data imputation and outlier detection, these flaws can negatively impact our prediction. We also realize that some time-series features packages are very US-centric. For example, one of the countries in our dataset, Qatar, does not celebrate US holidays like Christmas. The seasons in Australia are also different because Australia is in the Southern hemisphere, and Qatar and Singapore are tropical countries that do not have a real winter season. These challenges restrict the number of features that we are able to extract from the time series data, and inserting features based on our limited understanding of geography, climate and religion in these countries may lead to additional bias.

Next steps:

Modeling:

- Base model: linear regression, disregarding the time variable
- Classical machine learning models: LGBM, XGBoost or CatBoost, disregarding the time variable
- Time series prediction: ARIMA (remove the data before vaccine came out), prophet
- Neural Network: LSTM
- Metrics: RMSE

Data Visualization:

- To build a dashboard to visualize the new-confirmed data, new-deceased and vaccination data.

Conclusion:

For data preparation, we dropped columns that contain too much missing value or do not provide much useful information. Then we imputed the missing value in remaining columns, customizing imputation methods based on the form of data (daily, cumulative, whether it changes with time). We also found and dropped outliers in feature new_deceased, new_persons_fully_vaccinated, and mobility_workplaces.

We then adopt feature engineering including autocorrelation and partial correlation, lags features, rolling window statistics, and datetime features. These feature engineering methods will help us better analyze and forecast for time series data.

Last but not least, we plan for our next steps. We will explore more modeling in the next phase: we will try model focus on time series like ARIA and LSTM. We will also try classical learning models like Boosting, while disregarding the time variable.