Class Project

Instructor

Dr. Apirak Hoonlor

Dr. Akara Supratak

Submitted by

Mr. Varit Rungbanapan 6288039

Mr. Waipop Permpornskul 6288104

Miss Rungrawee Akkarapattanakoon 6288115

ITCS498 Special Topics in Computer Science

Mahidol University, Faculty of Information and Communication Technology

Semester 2 Academic Year 2021

Table of Contents

Introduction 3
Literature review and related work 4
Insight Finding 5
Data Management for Insight Finding 5
The interesting insights 7
Build the Models 13
The idea of the model 13
Data Management for Model Building 16
ARIMA 18
Auto Regression 24
Random Forest Regression and Linear Regression 29
Comparison 33
Evaluation With the Real Data 38
Discussion 41
Conclusion 42

Introduction

The precious lifestyle of everyone in the world has to be changed due to the unexpected Covid-19 crisis in 2019. People have to keep their distance from each other as well as wear masks, wash their hands, and receive the vaccine in order to go outside. Moreover, there are many numbers of pathetic deceased of coronavirus which results in fear and despair for many alive people. Although some people might not die from the Covid-19 infection, they still receive several negative effects on their bodies such as the lung or the respiratory system. In efforts to end the pandemic, we decided to use collected data from the "Our World in Data" Covid-19 Dataset to find the interesting insights from the data and to create a time forecasting machine learning model for England starting from 31st January 2020 to 10th March 2022. We hope that the results from our analysis could play a part in alleviating the Covid-19 situation.

Literature review and related work

Coronavirus disease, also known as COVID-19, is an infectious disease caused by the SARS-CoV-2 virus. The first epidemic was reported in November 2019 in Wuhan, China. The Covid-19 epidemic has transformed the way we live throughout the world since it began in 2019. The number of people infected with the SARS-CoV-2 virus had reached 388,188,172 as of February 4th, 2022. Various public and private organizations in many countries have collected and made the Covid-19 data public in an effort to end the pandemic. The "Our World in Data" Covid-19 Dataset is one of the worldwide repositories. In “Our World in Data” Covid-19 Dataset, can divide data into 8 metrics which are Vaccinations, Tests & positivity, Hospital & ICU, Confirmed cases, Confirmed deaths, Reproduction rate, and Policy responses, and Other variables of interest.

To do a model evaluation, there are many models that can be used. The model can be decided based on the data set and predicted the value that we have and also the try and trial.

Autoregression, or AR, is a time series model that predicts the future values based on past values as input to a regression equation.

Autoregressive Integrated Moving Average, or ARIMA, is a statistical analysis model that uses time-series data to either better understand the data set or to predict future trends. It consists of 2 terms which are AR (Auto regressive) and MA (Moving Average). Therefore, it is a form of regression analysis that determines how strong one dependent variable is in comparison to other changing variables.

Long Short-Term Memory models, or LSTM, are powerful time-series models. They have the ability to forecast any number of steps in the future. An LSTM module (or cell) comprises five key components that enable it to model both long and short-term data.

A linear regression model describes the relationship between one or more independent variables (x), and a dependent variable (y). A linear relationship between the input variables (x) and the output variable (y), y can be calculated using a linear combination of the input variables(x).

A Random Forest is an ensemble technique that uses multiple decision trees and a technique called Bootstrap and Aggregation to perform both regression and classification tasks. The main idea is to aggregate numerous decision trees to determine the final output. As a basic learning model, Random Forest uses several decision trees. It randomly performs row sampling and feature sampling from the dataset to create sample datasets for each model.

Insight Finding

Data Management for Insight Finding

** We have to separate the data preparation into two parts which is Data Management for Insight Finding and Data Management for Model Building because scope of the data that we use for insight finding and model building is different.

1. Before starting, we have to check what data we have in order to decide what to do next.

```
df.columns
```

```
Index(['iso_code', 'continent', 'location', 'date', 'total_cases', 'new_cases',
       'new_cases_smoothed', 'total_deaths', 'new_deaths',
       'new_deaths_smoothed', 'total_cases_per_million',
       'new_cases_per_million', 'new_cases_smoothed_per_million',
       'total_deaths_per_million', 'new_deaths_per_million',
       'new_deaths_smoothed_per_million', 'reproduction_rate', 'icu_patients',
       'icu_patients_per_million', 'hosp_patients',
       'hosp_patients_per_million', 'weekly_icu_admissions',
       'weekly_icu_admissions_per_million', 'weekly_hosp_admissions',
       'weekly_hosp_admissions_per_million', 'new_tests', 'total_tests',
       'total_tests_per_thousand', 'new_tests_per_thousand',
       'new_tests_smoothed', 'new_tests_smoothed_per_thousand',
       'positive_rate', 'tests_per_case', 'tests_units', 'total_vaccinations',
       'people_vaccinated', 'people_fully_vaccinated', 'total_boosters',
       'new_vaccinations', 'new_vaccinations_smoothed',
       'total_vaccinations_per_hundred', 'people_vaccinated_per_hundred',
       'people_fully_vaccinated_per_hundred', 'total_boosters_per_hundred',
       'new_vaccinations_smoothed_per_million',
       'new_people_vaccinated_smoothed',
       'new_people_vaccinated_smoothed_per_hundred', 'stringency_index',
       'population', 'population_density', 'median_age', 'aged_65_older',
       'aged_70_older', 'gdp_per_capita', 'extreme_poverty',
       'cardiovasc_death_rate', 'diabetes_prevalence', 'female_smokers',
       'male_smokers', 'handwashing_facilities', 'hospital_beds_per_thousand',
       'life_expectancy', 'human_development_index',
       'excess_mortality_cumulative_absolute', 'excess_mortality_cumulative',
       'excess_mortality', 'excess_mortality_cumulative_per_million'],
      dtype='object')
```

2. We use DataFrame called “clean_df” to store our clean data. First of all, we will drop unnecessary columns which we will not use them.

```
clean_df = df.copy()
clean_df = clean_df.drop(columns = ['location', 'new_cases_smoothed', 'new_deaths_smoothed',
                                    'total_cases_per_million','new_cases_per_million','new_cases_smoothed_per_million',
                                    'total_deaths_per_million','new_deaths_per_million','new_deaths_smoothed_per_million',
                                    'icu_patients_per_million','hosp_patients_per_million','weekly_icu_admissions_per_million',
                                    'weekly_hosp_admissions_per_million','total_tests_per_thousand','new_tests_per_thousand',
                                    'new_tests_smoothed_per_thousand','new_vaccinations_smoothed',
                                    'total_vaccinations_per_hundred','people_vaccinated_per_hundred',
                                    'people_fully_vaccinated_per_hundred','total_boosters_per_hundred',
                                    'new_vaccinations_smoothed_per_million','new_people_vaccinated_smoothed',
                                    'new_people_vaccinated_smoothed_per_hundred','aged_70_older','life_expectancy',
                                    'new_tests_smoothed','excess_mortality_cumulative_per_million',
                                    'excess_mortality_cumulative_absolute'
                                    ])
```

3. After dropping unneeded columns, we have to deal with the null value, so we have 3 methods to deal with it including changing it to 1 or 0 for the test_units column, filling the null value with 0, and filling the null value with the mean.

```
clean_df['tests_units'] = clean_df['tests_units'].notnull().astype("int")
```

```
clean_df['total_cases'] = clean_df['total_cases'].fillna(0)
clean_df['new_cases'] = clean_df['new_cases'].fillna(0)
clean_df['total_deaths'] = clean_df['total_deaths'].fillna(0)
clean_df['new_deaths'] = clean_df['new_deaths'].fillna(0)
clean_df['icu_patients'] = clean_df['icu_patients'].fillna(0)
clean_df['hosp_patients'] = clean_df['hosp_patients'].fillna(0)
clean_df['weekly_icu_admissions'] = clean_df['weekly_icu_admissions'].fillna(0)
clean_df['weekly_hosp_admissions'] = clean_df['weekly_hosp_admissions'].fillna(0)
clean_df['new_tests'] = clean_df['new_tests'].fillna(0)
clean_df['total_tests'] = clean_df['total_tests'].fillna(0)
clean_df['total_vaccinations'] = clean_df['total_vaccinations'].fillna(0)
clean_df['people_vaccinated'] = clean_df['people_vaccinated'].fillna(0)
clean_df['people_fully_vaccinated'] = clean_df['people_fully_vaccinated'].fillna(0)
clean_df['total_boosters'] = clean_df['total_boosters'].fillna(0)
clean_df['new_vaccinations'] = clean_df['new_vaccinations'].fillna(0)
clean_df['population'] = clean_df['population'].fillna(0)
clean_df['population_density'] = clean_df['population_density'].fillna(0)
clean_df['aged_65_older'] = clean_df['aged_65_older'].fillna(0)
clean_df['gdp_per_capita'] = clean_df['gdp_per_capita'].fillna(0)
clean_df['cardiovasc_death_rate'] = clean_df['cardiovasc_death_rate'].fillna(0)
clean_df['diabetes_prevalence'] = clean_df['diabetes_prevalence'].fillna(0)
clean_df['extreme_poverty'] = clean_df['extreme_poverty'].fillna(0)
clean_df['female_smokers'] = clean_df['female_smokers'].fillna(0)
clean_df['male_smokers'] = clean_df['male_smokers'].fillna(0)
clean_df['handwashing_facilities'] = clean_df['handwashing_facilities'].fillna(0)
clean_df['hospital_beds_per_thousand'] = clean_df['hospital_beds_per_thousand'].fillna(0)
clean_df['human_development_index'] = clean_df['human_development_index'].fillna(0)
```

```
clean_df['reproduction_rate'].fillna(clean_df.reproduction_rate.mean(),inplace=True)
clean_df['median_age'].fillna(clean_df.median_age.mean(),inplace=True)
clean_df['positive_rate'].fillna(clean_df.positive_rate.mean(),inplace=True)
clean_df['tests_per_case'].fillna(clean_df.tests_per_case.mean(),inplace=True)
clean_df['stringency_index'].fillna(clean_df.stringency_index.mean(),inplace=True)
clean_df['excess_mortality_cumulative'].fillna(clean_df.excess_mortality_cumulative.mean(),inplace=True)
clean_df['excess_mortality'].fillna(clean_df.excess_mortality.mean(),inplace=True)
```

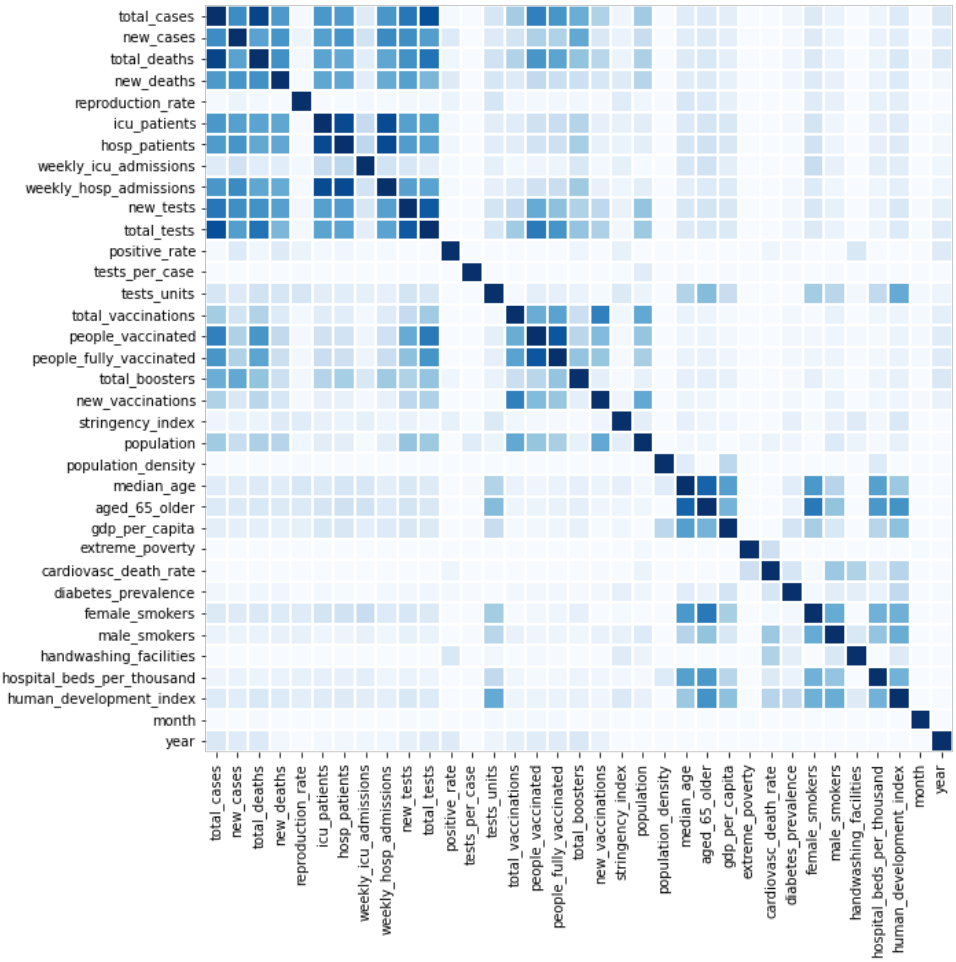
4. The preparation is already finished, and this is an example of the result.

```
clean_df
```

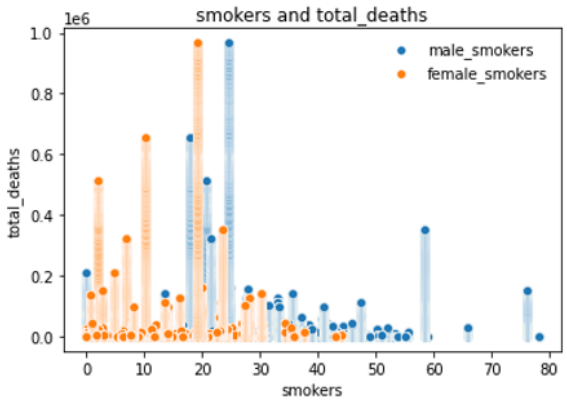
	iso_code	continent	date	total_cases	new_cases	total_deaths	new_deaths	reproduction_rate	icu_patients	hosp_patients	...	diabetes_prevalence
0	AFG	Asia	2020-02-24	5.0	5.0	0.0	0.0	1.007747	0.0	0.0	...	9.59
1	AFG	Asia	2020-02-25	5.0	0.0	0.0	0.0	1.007747	0.0	0.0	...	9.59
2	AFG	Asia	2020-02-26	5.0	0.0	0.0	0.0	1.007747	0.0	0.0	...	9.59
3	AFG	Asia	2020-02-27	5.0	0.0	0.0	0.0	1.007747	0.0	0.0	...	9.59
4	AFG	Asia	2020-02-28	5.0	0.0	0.0	0.0	1.007747	0.0	0.0	...	9.59
...	...	...	...	...	...	...	...	...	...	...	...	...
162857	ZWE	Africa	2022-02-14	231603.0	222.0	5374.0	0.0	0.810000	0.0	0.0	...	1.82
162858	ZWE	Africa	2022-02-15	231603.0	0.0	5374.0	0.0	0.780000	0.0	0.0	...	1.82
162859	ZWE	Africa	2022-02-16	232213.0	610.0	5379.0	5.0	1.007747	0.0	0.0	...	1.82
162860	ZWE	Africa	2022-02-17	232598.0	385.0	5381.0	2.0	1.007747	0.0	0.0	...	1.82
162861	ZWE	Africa	2022-02-18	233030.0	432.0	5385.0	4.0	1.007747	0.0	0.0	...	1.82

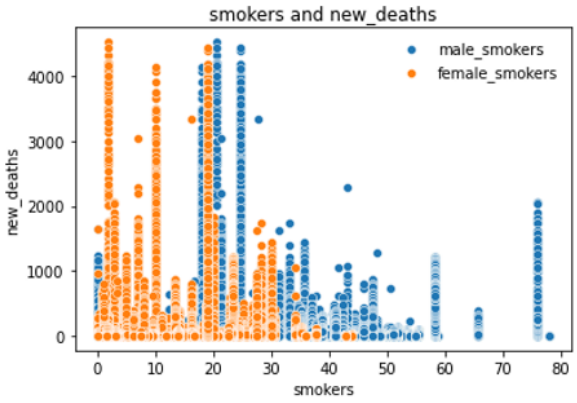
153101 rows × 40 columns

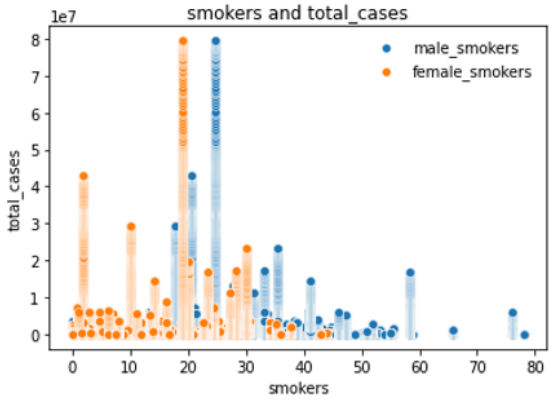
The interesting insights

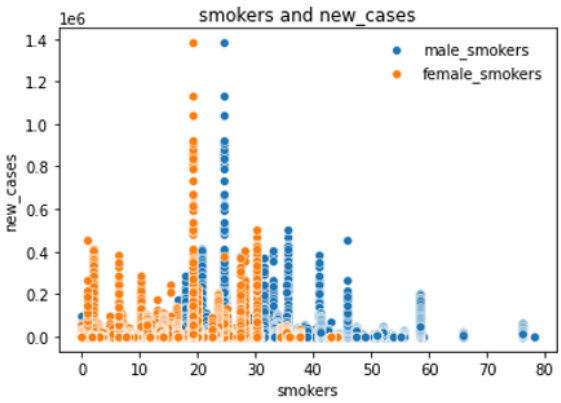


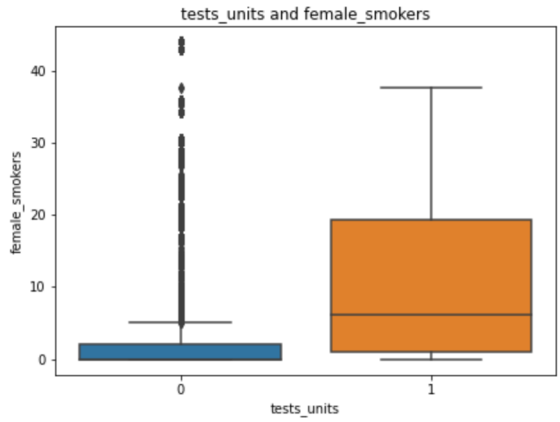
From the figure, there are correlation between each column of the Data Frame which is dark blue is high correlation and light blue is low correlation.

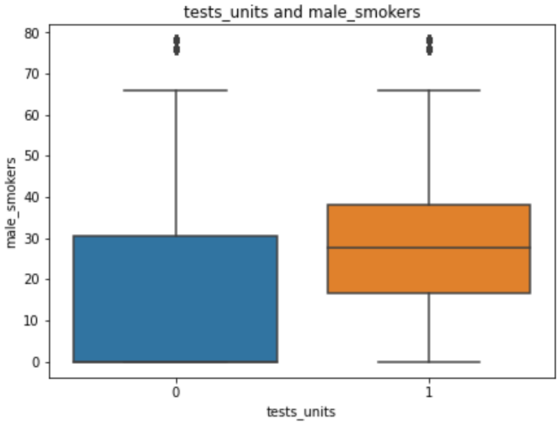




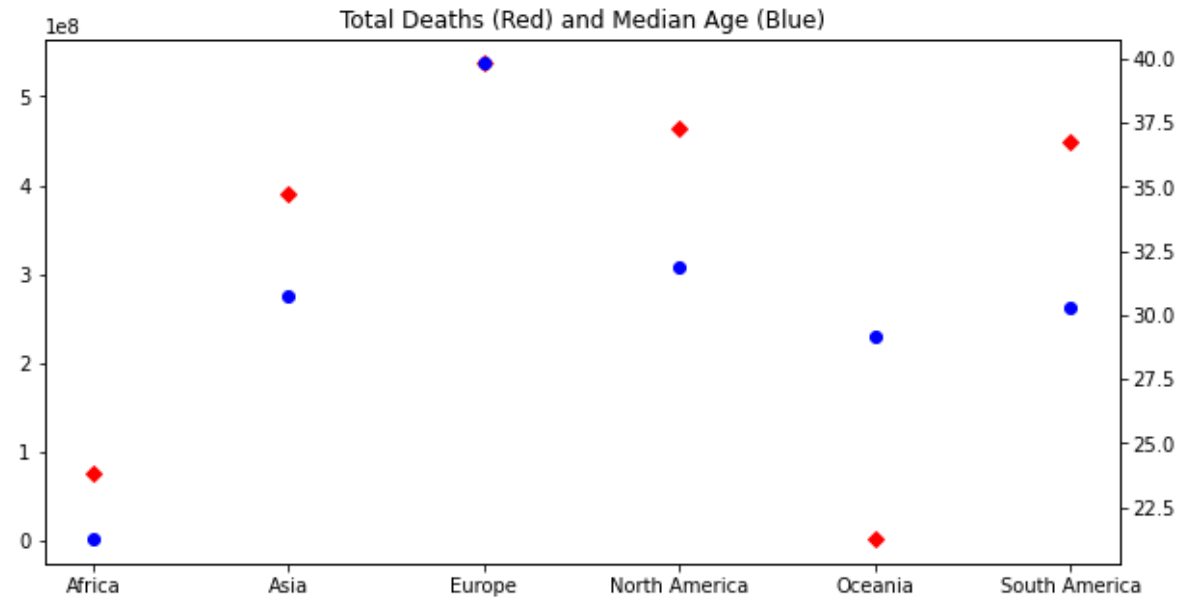




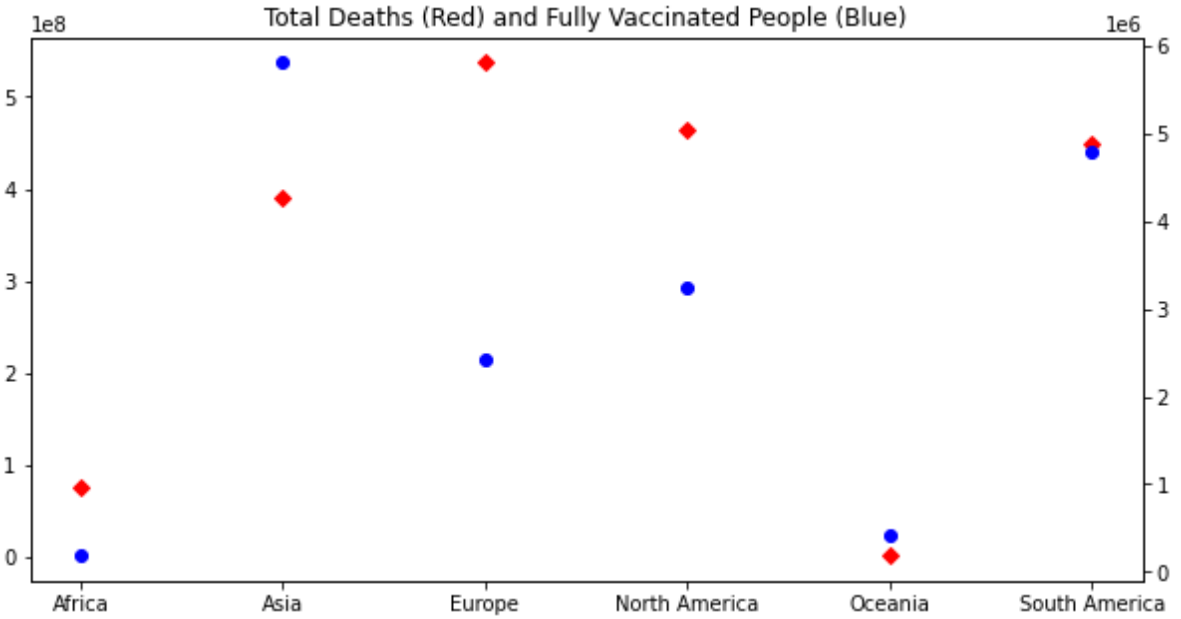




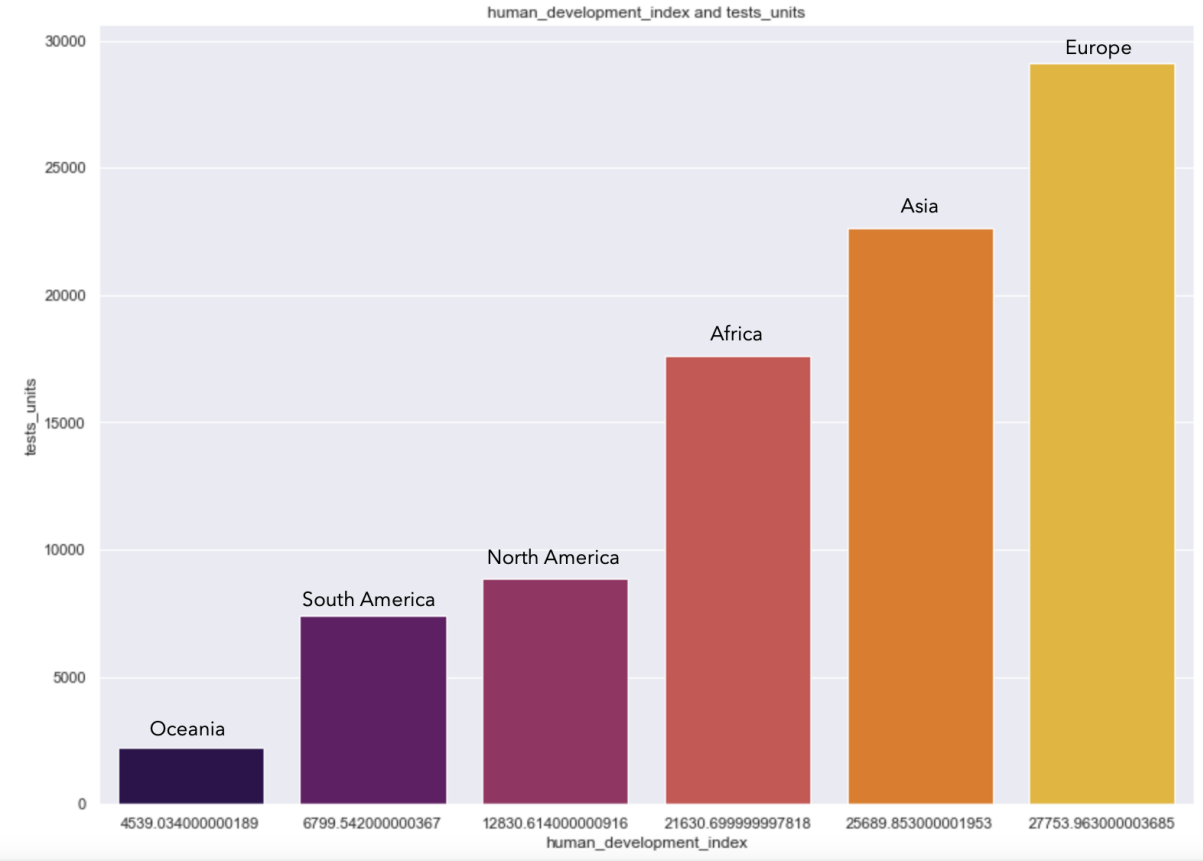
From observation, it found that the mean of female smokers less than male smokers and the rate of total deaths, total cases and new cases seems not effected by the rate of smokers. However, the rate of smokers is tending to be directly proportional to the performed tests units which is 1.



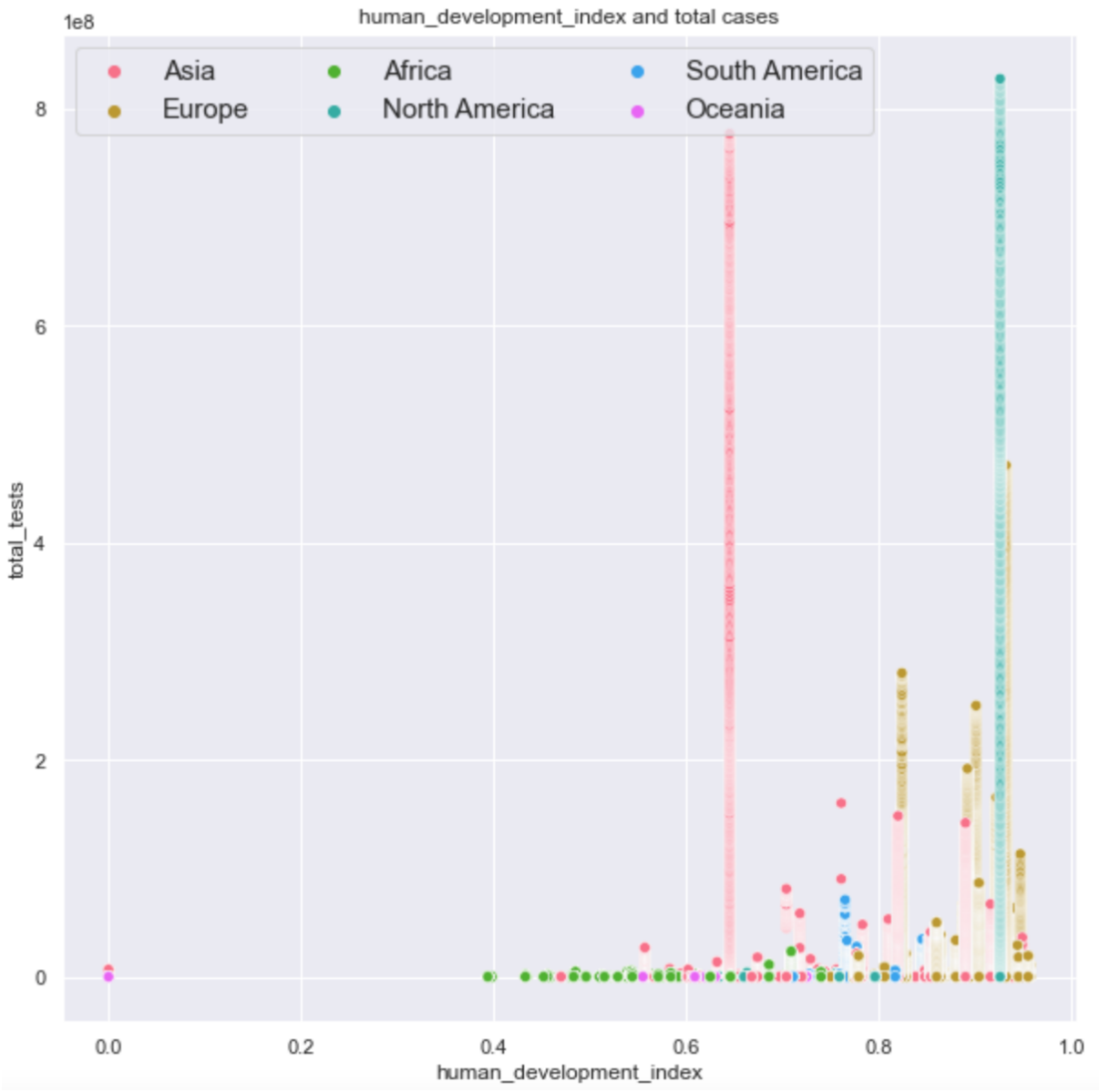
We can know from the graph that continents that have higher median age tend to have a higher number of dead people. For example, Europe, which has the highest median age, has the most deaths even though Europe has the second least population. Therefore, it can imply that older people are more likely to die from covid-19 than younger people.



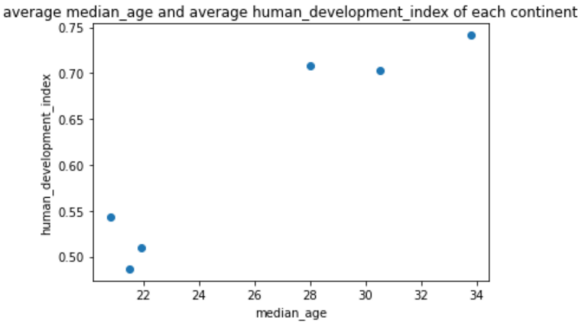
We can know from the graph that continents that have a higher number of fully vaccinated people tend to have a lower number of dead people. For example, Asia, which has the highest number of people who are fully vaccinated, has lower deaths than Europe even though Asia has the most population. Therefore, it can imply that vaccines can reduce the chance of dying from covid-19.

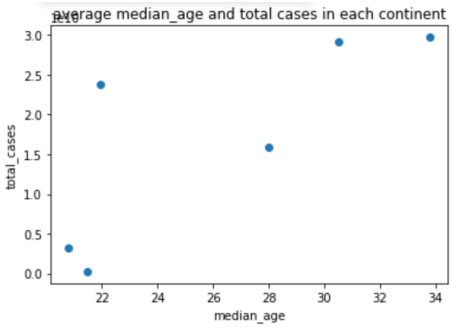


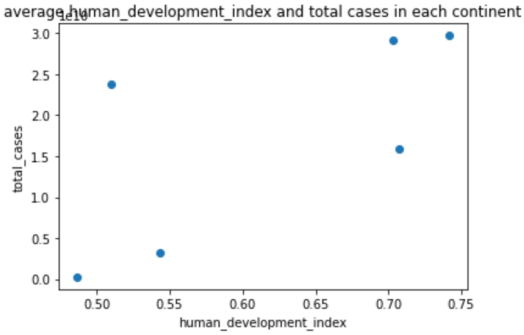
As we observed from the graph, we found that the continent which has a higher human development index will have more test units. It has been arranged in ascending order as Oceania, South America, North America, Africa, Asia, and Europe. As the Human Development Index is used to quantify a country's average achievement of human development which is a long and healthy life, knowledge, and decent standard of living, so we can conclude that the human development index is directly proportional to the number of test units. This means in a continent that has a high human development index, citizens will have more chances to access test units.



Also, the continent which has a higher human development index will have more total tests. By the way, the number of total tests also depends on the population of each continent. For example, Asia is at a moderate score on the human development index but has an extremely high amount of total tests.







From observation, it found that the human development index tends to be directly proportional to the average median age of each continent which can assume that the continent that has a higher average rate of median age has tended to have more cases. However, it seems that the higher human development index does not affect the awareness of the covid-19 in those continents based on the total cases is seem to be directly proportional to the human development index.

Build the Models

The idea of the model

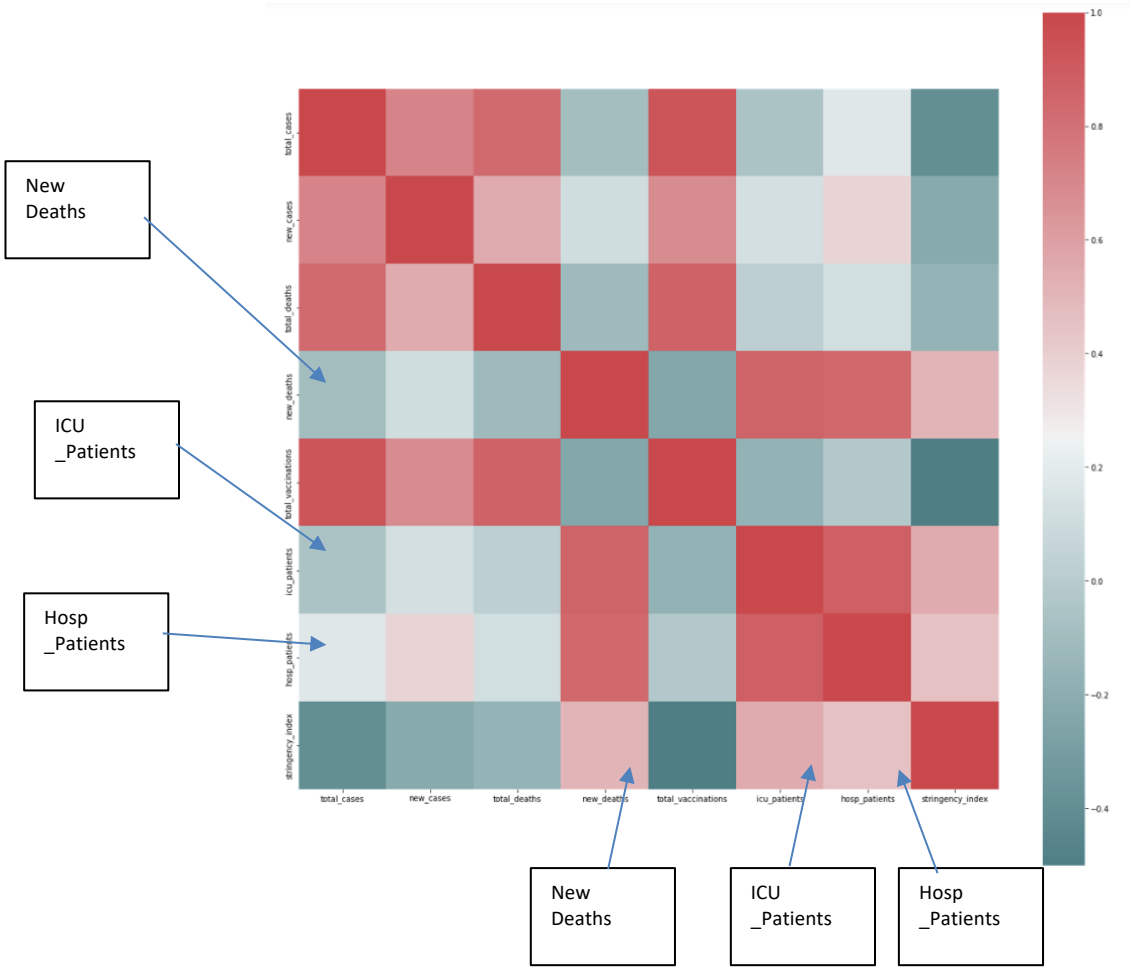
What did we want to create?

In this project, we have the idea to create a time forecasting machine learning model about the number of hospital patients, ICU patients, and total deaths in England. It would be great if the related organization can predict the amount of these columns.

Why have we used only these three columns of England?

There are many columns that could be used in the dataset, but these three columns are useful in terms of medical field management. Hospitals and related organizations will get benefit from the bed and room preparation for the covid-19 patients in the future. Another reason is that hospital patients, ICU patients, and new deaths have a strong relationship with each other following the heat map graph below. The reason why we choose England as the country is that the England dataset was almost completed with a few null values which are good for the machine learning model.

- The date that we use for model creation is 31st January 2020 to 10th March 2022
- We separate the last 30 days dataset out for the test dataset.



Plan for model creating.

1. Explore the dataset and find the useful columns that can be used for model creation.
2. Find the Country that has satisfied the dataset. (The country that has just a few null values in the columns that we are going to use.)
3. Do Brainstorming with the team about what is the model that we can do for the data.
4. Data Management (Clean the data and pick the columns that we want to use)
5. Research for the method that relates to what we are going to do.
6. Try to implement the methods and test the performance.
7. Compare the performance of the models that we created by using different methods.
8. Analyze and do the conclusion about the models.

The models that we tried to use.

1. ARIMA
2. AUTO REGRESSION
3. Deep Learning Model using LSTM (Failed during development)
4. Linear Regression
5. Random Forest Regression

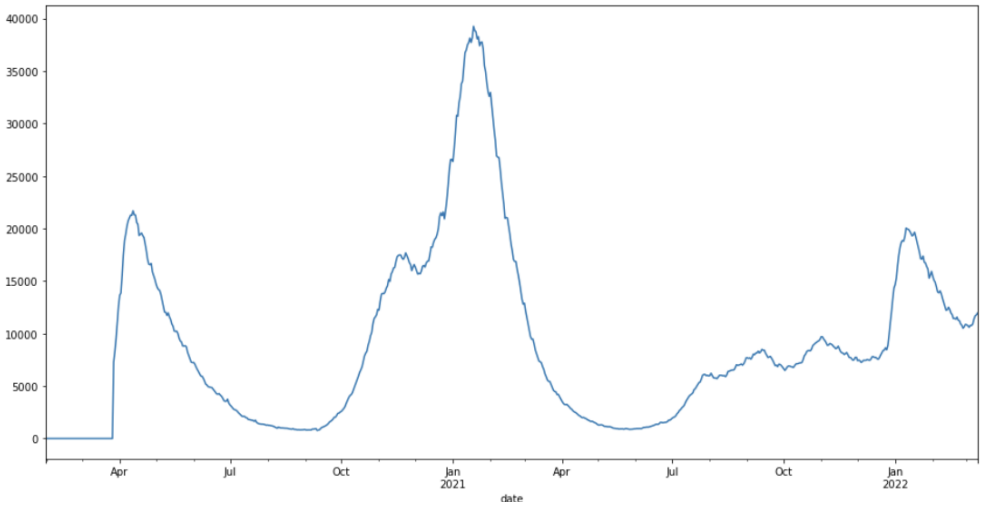
Prediction of the model's performance before start.

ARIMA and Auto Regression will give a better prediction than linear regression and Random Forest Regression because they are the model for time forecasting. Random Forest and Linear Regression might perform better if they are optimized.

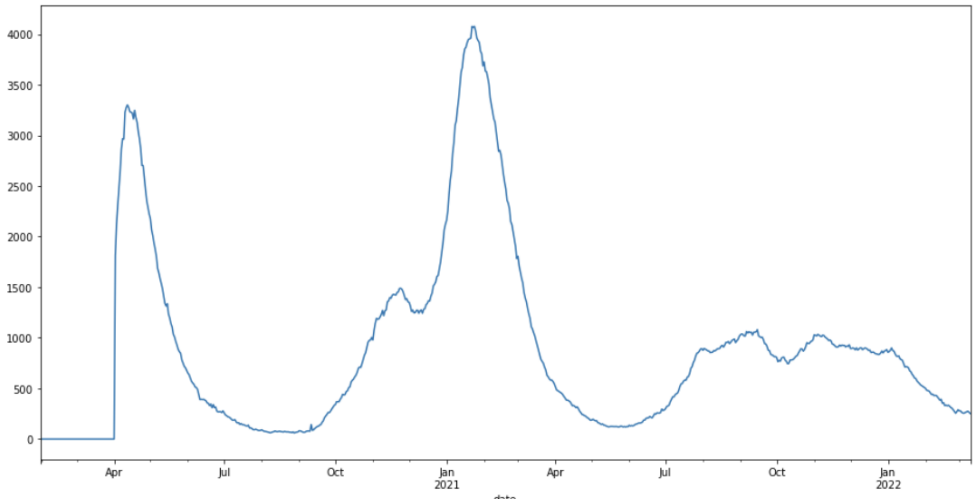
The example of the dataset that we are going to use.

date	hosp_patients	icu_patients	total_deaths
2020-01-31	0.0	0.0	0.0
2020-02-01	0.0	0.0	0.0
2020-02-02	0.0	0.0	0.0
2020-02-03	0.0	0.0	0.0
2020-02-04	0.0	0.0	0.0
...	...	...	...
2022-03-06	10902.0	263.0	162152.0
2022-03-07	11368.0	269.0	162292.0
2022-03-08	11678.0	275.0	162505.0
2022-03-09	11773.0	266.0	162628.0
2022-03-10	11944.0	253.0	162770.0

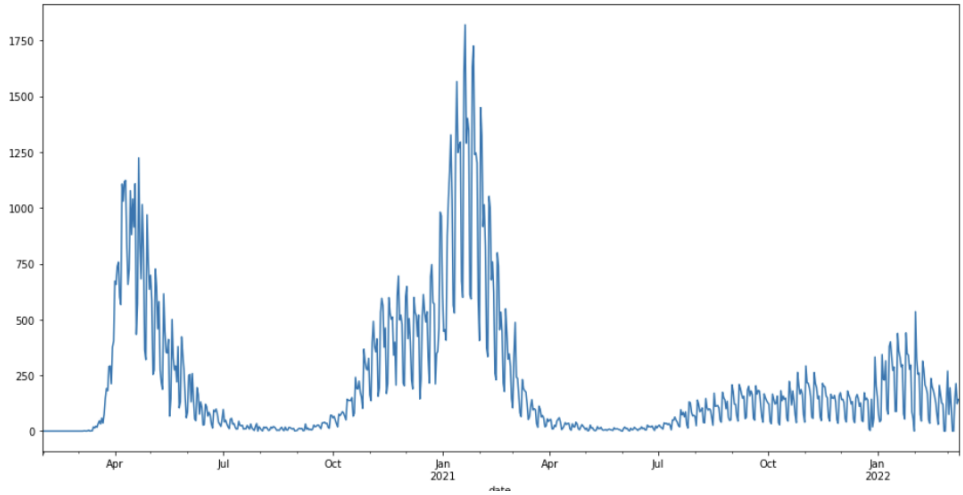
770 rows × 3 columns



The graph of the total hospital patients between 31st January 2020 to 10th March 2022



The graph of the total ICU patients between 31st January 2020 to 10th March 2022



The graph of the new deaths between 31st January 2020 to 10th March 2022

Data Management for Model Building

In the model building section, we decided to build the model about the Great Britain. You can find more information about it in **"Why have we used only these three columns of England?"** section

Step to do the data management for model building:

1. Pick the columns that we might use for model building.
(Hospital Patients, ICU Patients, and New deaths are the main columns that we are going to use for sure, but other columns are interesting and have potential to be the columns that we might use. From that reason, we decided to clean every column that we might use in this step.)

2. Select the country that does not contain many null values in the columns that we chose. (We decided to use the dataset of Great Britain)

```
In [257]: eng_df = Europe_df.loc[Europe_df['iso_code'] == 'GBR'].reset_index()


          eng_model_df = eng_df[['date','total_cases','new_cases','total_deaths'
                                 ,'new_deaths','total_vaccinations','icu_patients','hosp_patients']]
          eng_model_df.drop(eng_model_df.index[[770,771]], inplace=True)
```

Select the country and pick the interesting columns

3. Change the type of "date" column into datetime64 type.

```
In [12]: eng_model_df['date'] = pd.to_datetime(eng_model_df['date'], format='%Y-%m-%d')
```

Change the type of date column

4. Since total_deaths, new_deaths, total_vaccinations, icu_patients, and hosp_patients, contain null value on the first rows that there is no number on that days, its mean that the umber should be zero because there is no case. (For Example, the first 10 days of total_deaths are null because there was no death on the first 10 days, so replace zero instead of null is reasonable.)

```
|: eng_use_df['total_deaths'] = eng_use_df['total_deaths'].fillna(0)
   eng_use_df['new_deaths'] = eng_use_df['new_deaths'].fillna(0)
   eng_use_df['total_vaccinations'] = eng_use_df['total_vaccinations'].fillna(0)
   eng_use_df['icu_patients'] = eng_use_df['icu_patients'].fillna(0)
   eng_use_df['hosp_patients'] = eng_use_df['hosp_patients'].fillna(0)
```

Replace zero in some columns

5.There are not that much null values in new cases column. It is reasonable to replace the null value with the adjacent data.

```
In [16]: eng_model_df['new_cases'] = eng_model_df['new_cases'].fillna(method="ffill")
```

Replace null value with adjacent data

6. Check the type of and number of null values in each column.

```
In [261]: eng_model_df.dtypes
Out[261]: date                 datetime64[ns]
          total_cases                 float64
          new_cases                   float64
          total_deaths                float64
          new_deaths                  float64
          total_vaccinations          float64
          icu_patients                float64
          hosp_patients               float64
          dtype: object
```

Check the type of each column

```
In [265]: eng_model_df.isnull().sum()
Out[265]: date                 0
          total_cases          0
          new_cases            0
          total_deaths         0
          new_deaths           0
          total_vaccinations   0
          icu_patients         0
          hosp_patients        0
          dtype: int64
```

Check the null value of each column

ARIMA

We decided to try ARIMA for the model creation because we found that it is the method for the time forecasting, so we thought that it is going to perform well.

Step of the model creating using ARIMA.

1. Get the stats of the dataset using adfuller

   Before start creating the model, it would be great if we know the stats of the dataset that we are going to use.

```
In [29]: ad_test(Forecasting_df['hosp_patients'])
         1. ADF :  -2.834502611501363
         2. P-Value :  0.05351182092885613
         3. Num Of Lags :  20
         4. Num Of Observations Used For ADF Regression: 749
         5. Critical Values :
                 1% :  -3.439110818166223
                 5% :  -2.8654065210185795
                 10% :  -2.568828945705979
```

The stats of hospital patients dataset

```
1. ADF :  -3.4321507722519526
2. P-Value :  0.009910479355914268
3. Num Of Lags :  8
4. Num Of Observations Used For ADF Regression: 761
5. Critical Values :
        1% :  -3.4389722010249386
        5% :  -2.8653454308425705
        10% :  -2.5687964010457227
```

The stats of ICU patients dataset

```
1. ADF :  -3.754283774224442
2. P-Value :  0.0034112556964436026
3. Num Of Lags :  16
4. Num Of Observations Used For ADF Regression: 753
5. Critical Values :
        1% :  -3.4390641198617864
        5% :  -2.8653859408474482
        10% :  -2.5688179819544312
```

The stats of the New Deaths dataset

2. Find the best ARIMA Parameter for the dataset

```
In [30]: from pmdarima import auto_arima
         stepwise_fit = auto_arima(Forecasting_df['hosp_patients'], trace=True,
         suppress_warnings=True)

         Performing stepwise search to minimize aic
          ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=11318.309, Time=0.35 sec
          ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=11639.529, Time=0.02 sec
          ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=11428.061, Time=0.03 sec
          ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=11506.895, Time=0.12 sec
          ARIMA(0,1,0)(0,0,0)[0]             : AIC=11638.380, Time=0.01 sec
          ARIMA(1,1,2)(0,0,0)[0] intercept   : AIC=11315.644, Time=0.38 sec
          ARIMA(0,1,2)(0,0,0)[0] intercept   : AIC=11453.473, Time=0.18 sec
          ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=11314.339, Time=0.28 sec
          ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=11315.719, Time=0.41 sec
          ARIMA(2,1,0)(0,0,0)[0] intercept   : AIC=11379.919, Time=0.05 sec
          ARIMA(1,1,1)(0,0,0)[0]             : AIC=11312.403, Time=0.10 sec
          ARIMA(0,1,1)(0,0,0)[0]             : AIC=11505.457, Time=0.07 sec
          ARIMA(1,1,0)(0,0,0)[0]             : AIC=11426.358, Time=0.02 sec
          ARIMA(2,1,1)(0,0,0)[0]             : AIC=11313.782, Time=0.15 sec
          ARIMA(1,1,2)(0,0,0)[0]             : AIC=11313.708, Time=0.16 sec
          ARIMA(0,1,2)(0,0,0)[0]             : AIC=11451.894, Time=0.10 sec
          ARIMA(2,1,0)(0,0,0)[0]             : AIC=11378.101, Time=0.03 sec
          ARIMA(2,1,2)(0,0,0)[0]             : AIC=11316.354, Time=0.32 sec

         Best model:  ARIMA(1,1,1)(0,0,0)[0]
         Total fit time: 2.799 seconds
```

Finding the appropriate Parameter for hospital patients' dataset

```
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=8740.576, Time=0.61 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=8871.634, Time=0.01 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=8803.245, Time=0.05 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=8823.054, Time=0.11 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=8869.648, Time=0.01 sec
 ARIMA(1,1,2)(0,0,0)[0] intercept   : AIC=8742.469, Time=0.28 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=8742.469, Time=0.25 sec
 ARIMA(3,1,2)(0,0,0)[0] intercept   : AIC=8744.383, Time=0.73 sec
 ARIMA(2,1,3)(0,0,0)[0] intercept   : AIC=8744.295, Time=0.76 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=8740.474, Time=0.16 sec
 ARIMA(0,1,2)(0,0,0)[0] intercept   : AIC=8799.949, Time=0.15 sec
 ARIMA(2,1,0)(0,0,0)[0] intercept   : AIC=8776.981, Time=0.05 sec
 ARIMA(1,1,1)(0,0,0)[0]             : AIC=8738.475, Time=0.08 sec
 ARIMA(0,1,1)(0,0,0)[0]             : AIC=8821.063, Time=0.06 sec
 ARIMA(1,1,0)(0,0,0)[0]             : AIC=8801.252, Time=0.03 sec
 ARIMA(2,1,1)(0,0,0)[0]             : AIC=8740.470, Time=0.12 sec
 ARIMA(1,1,2)(0,0,0)[0]             : AIC=8740.470, Time=0.15 sec
 ARIMA(0,1,2)(0,0,0)[0]             : AIC=8797.957, Time=0.08 sec
 ARIMA(2,1,0)(0,0,0)[0]             : AIC=8774.986, Time=0.03 sec
 ARIMA(2,1,2)(0,0,0)[0]             : AIC=inf, Time=0.42 sec

Best model:  ARIMA(1,1,1)(0,0,0)[0]
Total fit time: 4.147 seconds
```

Finding the appropriate Parameter for ICU patients' dataset

```
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=9604.722, Time=0.48 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=9906.520, Time=0.01 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=9904.149, Time=0.03 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=9822.908, Time=0.12 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=9904.522, Time=0.01 sec
 ARIMA(1,1,2)(0,0,0)[0] intercept   : AIC=9633.270, Time=0.31 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=9626.399, Time=0.42 sec
 ARIMA(3,1,2)(0,0,0)[0] intercept   : AIC=9606.721, Time=0.48 sec
 ARIMA(2,1,3)(0,0,0)[0] intercept   : AIC=9571.361, Time=0.69 sec
 ARIMA(1,1,3)(0,0,0)[0] intercept   : AIC=9465.642, Time=0.44 sec
 ARIMA(0,1,3)(0,0,0)[0] intercept   : AIC=9629.223, Time=0.35 sec
 ARIMA(1,1,4)(0,0,0)[0] intercept   : AIC=9422.815, Time=0.59 sec
 ARIMA(0,1,4)(0,0,0)[0] intercept   : AIC=9428.139, Time=0.48 sec
 ARIMA(2,1,4)(0,0,0)[0] intercept   : AIC=9525.022, Time=0.84 sec
 ARIMA(1,1,5)(0,0,0)[0] intercept   : AIC=9431.072, Time=0.95 sec
 ARIMA(0,1,5)(0,0,0)[0] intercept   : AIC=9420.428, Time=0.65 sec
 ARIMA(0,1,5)(0,0,0)[0]             : AIC=9418.436, Time=0.36 sec
 ARIMA(0,1,4)(0,0,0)[0]             : AIC=9426.150, Time=0.23 sec
 ARIMA(1,1,5)(0,0,0)[0]             : AIC=9419.112, Time=0.82 sec
 ARIMA(1,1,4)(0,0,0)[0]             : AIC=9420.824, Time=0.36 sec

Best model:  ARIMA(0,1,5)(0,0,0)[0]
Total fit time: 8.625 seconds
```

Finding the appropriate Parameter for the new deaths dataset

3. Train the model and get the stats of the model

We cut the data of the last 30 days out for the testing model and use the first 739 data

```
from statsmodels.tsa.arima_model import ARIMA
model=ARIMA(train['hosp_patients'],order=(1,1,1))
model=model.fit()
model.summary()
```

Dep. Variable:	D.hosp_patients	**No. Observations:**	739
Model:	ARIMA(1, 1, 1)	**Log Likelihood**	-5442.772
Method:	css-mle	**S.D. of innovations**	382.075
Date:	Wed, 11 May 2022	**AIC**	10893.544
Time:	16:51:16	**BIC**	10911.965
Sample:	02-01-2020	**HQIC**	10900.647
	- 02-08-2022		

	coef	std err	z	P>|z|	[0.025	0.975]
const	15.0678	74.564	0.202	0.840	-131.074	161.210
ar.L1.D.hosp_patients	0.9475	0.015	63.328	0.000	0.918	0.977
ma.L1.D.hosp_patients	-0.7158	0.030	-23.586	0.000	-0.775	-0.656

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	1.0554	+0.0000j	1.0554	0.0000
MA.1	1.3971	+0.0000j	1.3971	0.0000

Training and stats of the model of hospital patients

```
from statsmodels.tsa.arima_model import ARIMA
model=ARIMA(train['icu_patients'],order=(1,1,1))
model=model.fit()
model.summary()
```

ARIMA Model Results

Dep. Variable:	D.icu_patients	**No. Observations:**	739
Model:	ARIMA(1, 1, 1)	**Log Likelihood**	-4210.099
Method:	css-mle	**S.D. of innovations**	72.085
Date:	Wed, 11 May 2022	**AIC**	8428.197
Time:	16:51:21	**BIC**	8446.619
Sample:	02-01-2020	**HQIC**	8435.300
	- 02-08-2022		

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.4798	7.969	0.060	0.952	-15.139	16.099
ar.L1.D.icu_patients	0.9125	0.025	35.937	0.000	0.863	0.962
ma.L1.D.icu_patients	-0.7342	0.041	-17.987	0.000	-0.814	-0.654

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	1.0959	+0.0000j	1.0959	0.0000
MA.1	1.3620	+0.0000j	1.3620	0.0000

Training and stats of the model of ICU patients

```
from statsmodels.tsa.arima_model import ARIMA
model=ARIMA(train['new_deaths'],order=(1,1,5))
model=model.fit()
model.summary()
```

ARIMA Model Results

Dep. Variable:	D.new_deaths	**No. Observations:**	739
Model:	ARIMA(1, 1, 5)	**Log Likelihood**	-4522.050
Method:	css-mle	**S.D. of innovations**	109.701
Date:	Wed, 11 May 2022	**AIC**	9060.101
Time:	16:51:31	**BIC**	9096.943
Sample:	02-01-2020	**HQIC**	9074.307
	- 02-08-2022		

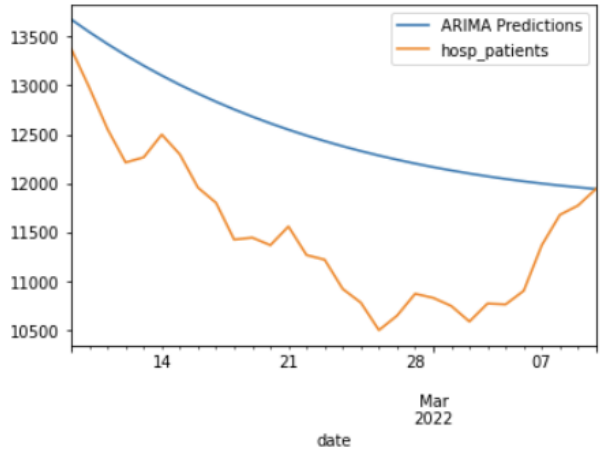
	coef	std err	z	P>|z|	[0.025	0.975]
const	0.2967	1.885	0.157	0.875	-3.398	3.991
ar.L1.D.new_deaths	-0.5759	0.097	-5.948	0.000	-0.766	-0.386
ma.L1.D.new_deaths	0.1413	0.092	1.538	0.124	-0.039	0.322
ma.L2.D.new_deaths	-0.9986	0.052	-19.203	0.000	-1.101	-0.897
ma.L3.D.new_deaths	-0.3807	0.079	-4.805	0.000	-0.536	-0.225
ma.L4.D.new_deaths	0.5423	0.031	17.698	0.000	0.482	0.602
ma.L5.D.new_deaths	0.4332	0.051	8.437	0.000	0.333	0.534

Roots

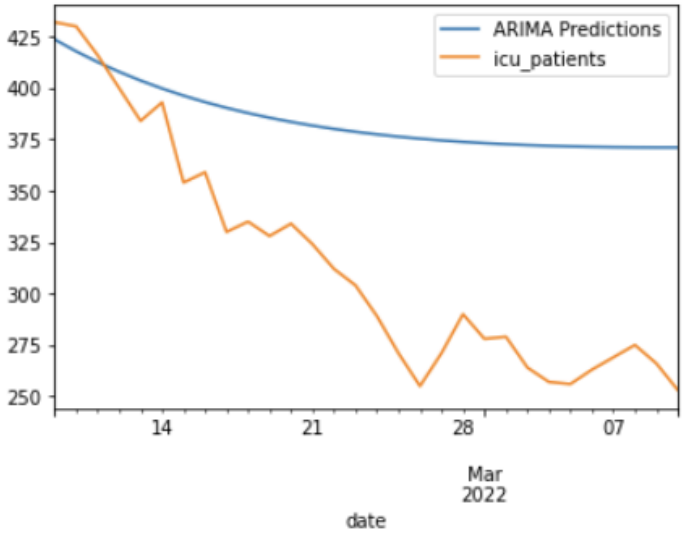
	Real	Imaginary	Modulus	Frequency
AR.1	-1.7364	+0.0000j	1.7364	0.5000
MA.1	0.9573	-0.4021j	1.0383	-0.0633
MA.2	0.9573	+0.4021j	1.0383	0.0633
MA.3	-0.9560	-0.8905j	1.3065	-0.3806
MA.4	-0.9560	+0.8905j	1.3065	0.3806
MA.5	-1.2545	-0.0000j	1.2545	-0.5000

Training and stats of the model of new deaths (We use 1,1,5 parameter because 0,1,5 cause the bug)

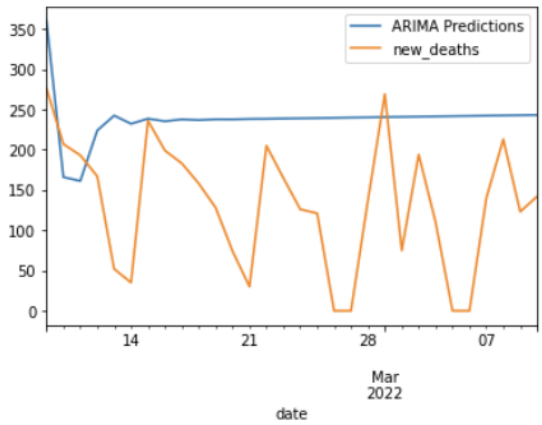
4. Test the model by comparing the prediction with the last 30 days of data which is the test dataset.



The comparison Graph between the prediction and test dataset (Last 30 days) of hospital Patients

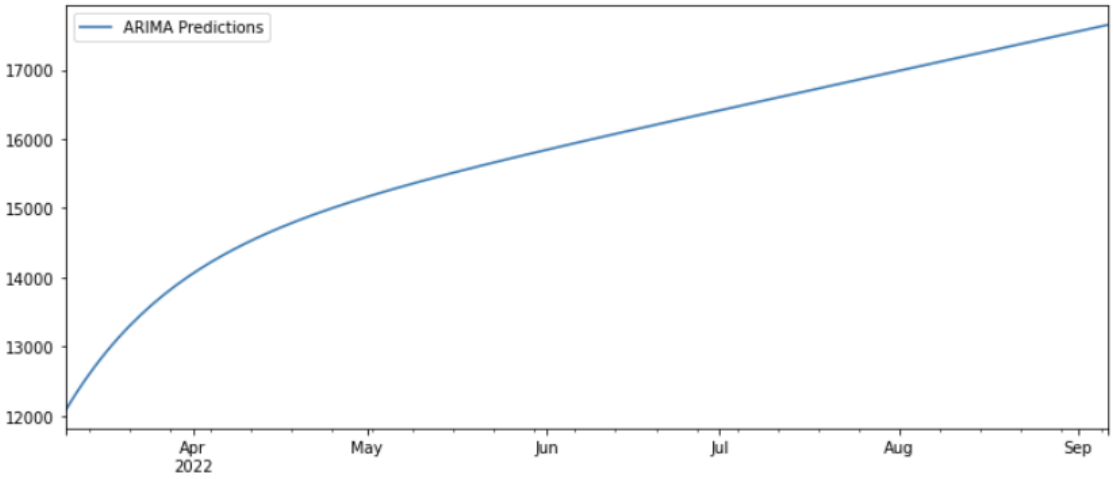


The comparison Graph between the prediction and test dataset (Last 30 days) of ICU Patients

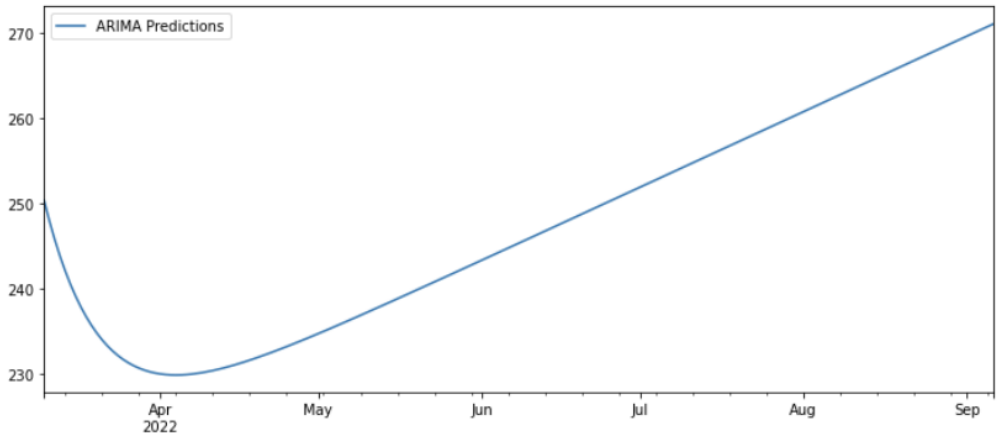


The comparison Graph between the prediction and test dataset (Last 30 days) of new deaths

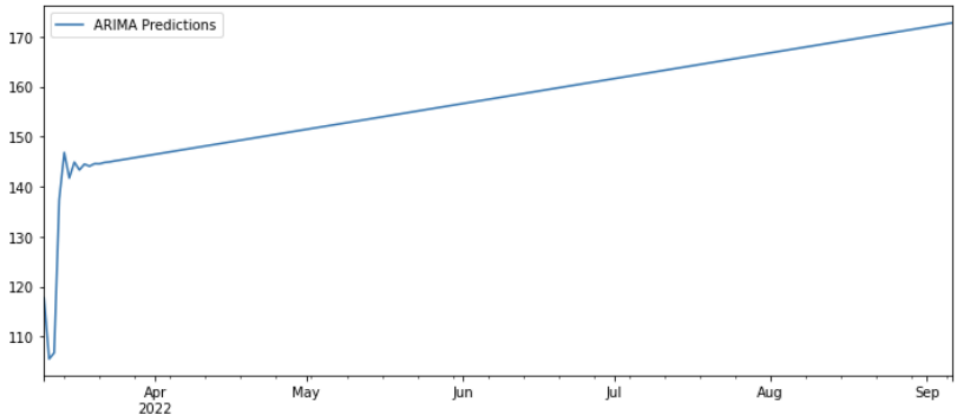
5. Make the prediction by set the last day of prediction to 6th September 2022



The Prediction of the hospital patients using ARIMA until 6th September 2022



The Prediction of the ICU patients using ARIMA until 6th September 2022



The Prediction of the new deaths using ARIMA until 6th September 2022

Conclusion of ARIMA Methods.

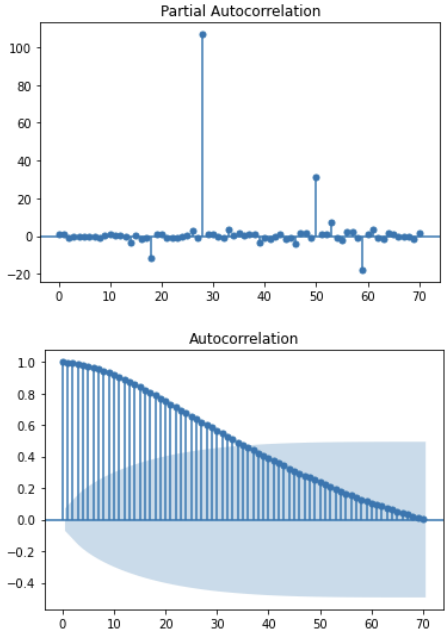
ARIMA is the method for time forecasting model creation, but it did not perform great in terms of accuracy and reasonable. The prediction of the last 30 days with the test dataset gives a different range of numbers which is not good. Moreover, the prediction until 6th September 2022 is unreasonable. This method might perform well if we know how to use it properly. The disadvantage point of this method is that this method id not flexible, so we cannot adjust many parameters or tune the model.

Auto Regression

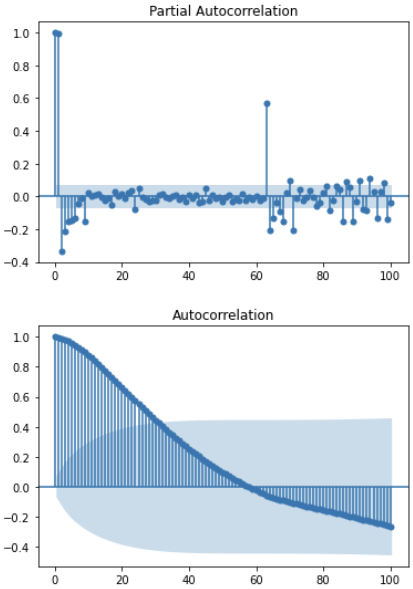
Auto Regression is the method to create the date forecasting model as ARIMA, after we get the result of the ARIMA model, we just need to know that will Auto Regression gives the poor result as ARIMA or not.

Step of the model creating using ARIMA.

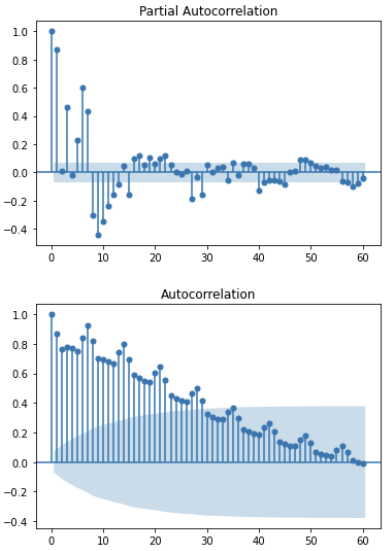
1. Prepare the dataset
2. Plot the PACF and ACF graphs to get the value of lags that are appropriate for the model.



The Partial Autocorrelation and Auto Correlation Graph of Hospital Patients



The Partial Autocorrelation and Auto Correlation Graph of ICU Patients



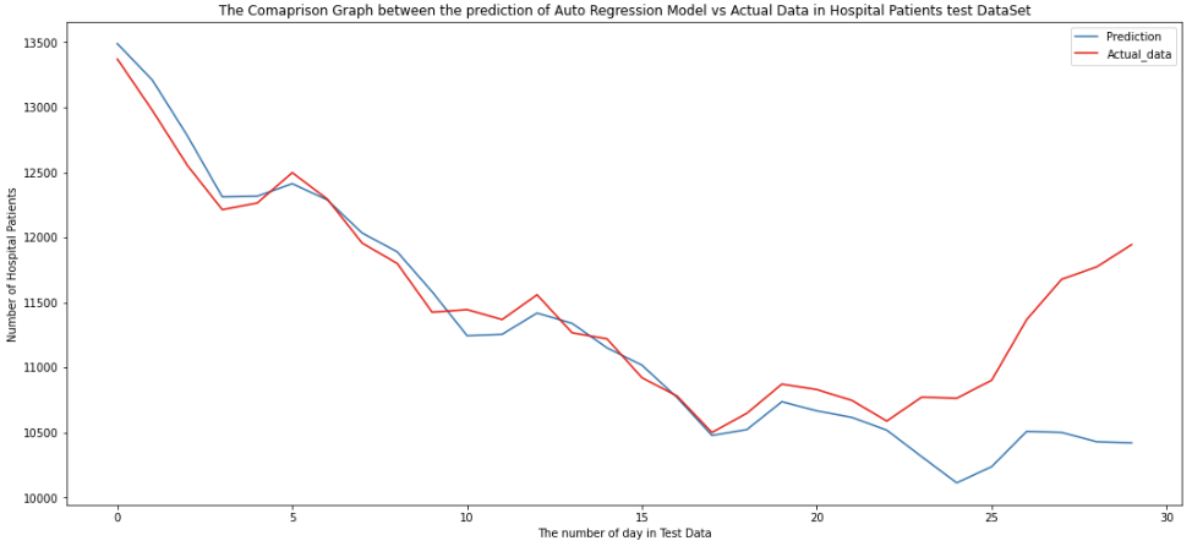
The Partial Autocorrelation and Auto Correlation Graph of New Deaths Patients

3. Train the model by separate last 30 days out to be the test data set

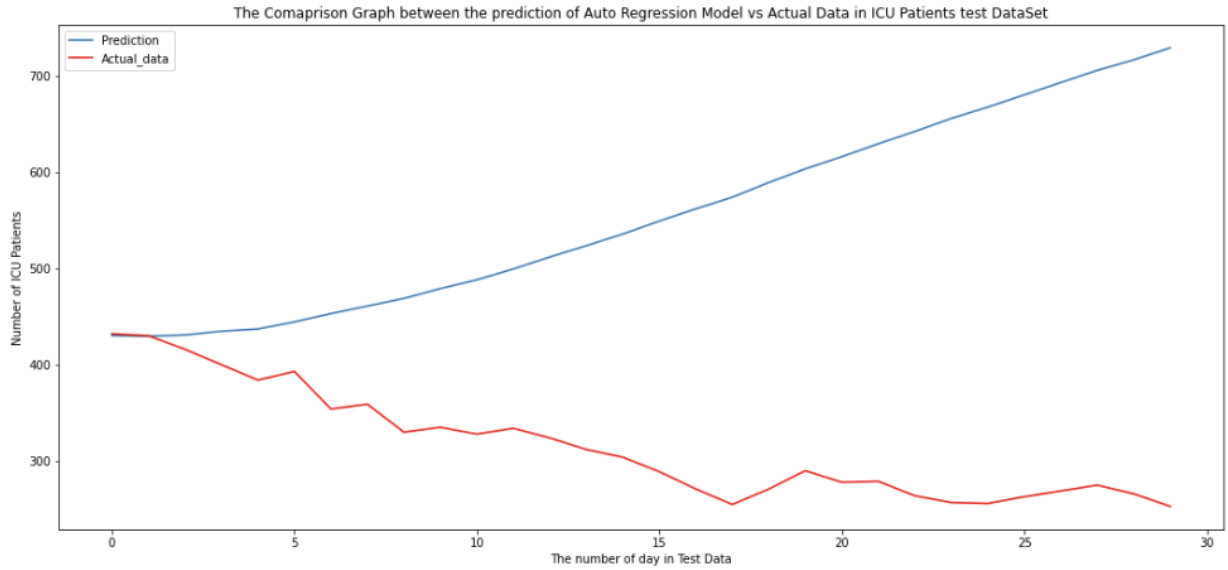
```
In [60]: print(Forecasting_df.shape)
         train=X[:len(X)-30]
         test=X[len(X)-30:]
         print(train.shape,test.shape)

         (770, 8)
         (740,) (30,)
```

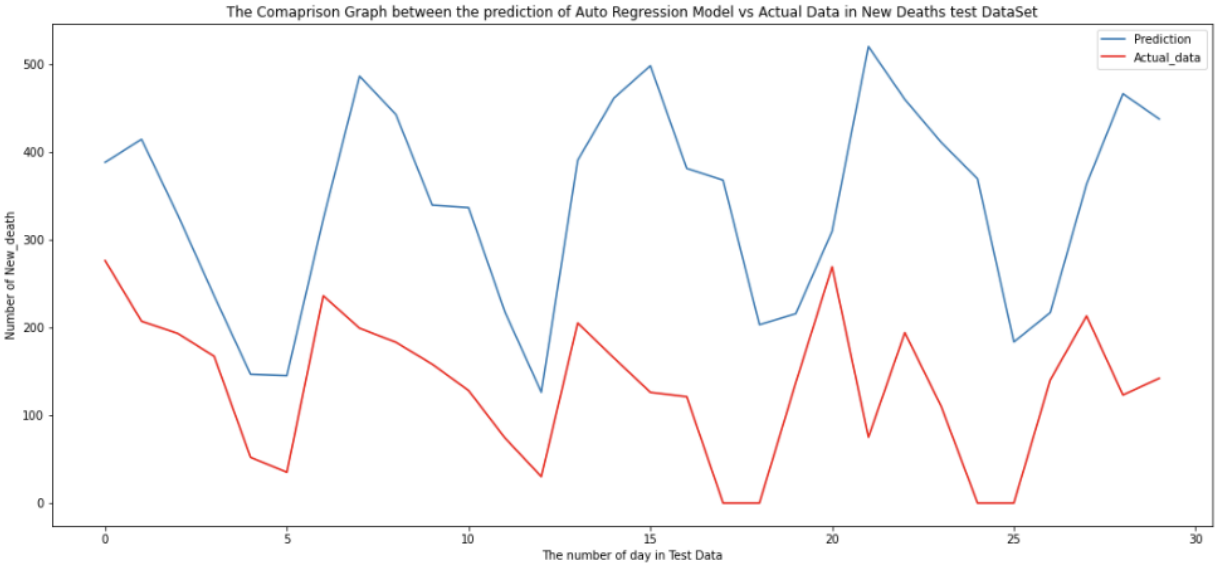
4. Compare the result of the last 30 days dataset with the prediction using Auto Regression



Prediction and the last 30 days of the dataset which is the test dataset of Hospital Patients Dataset comparison

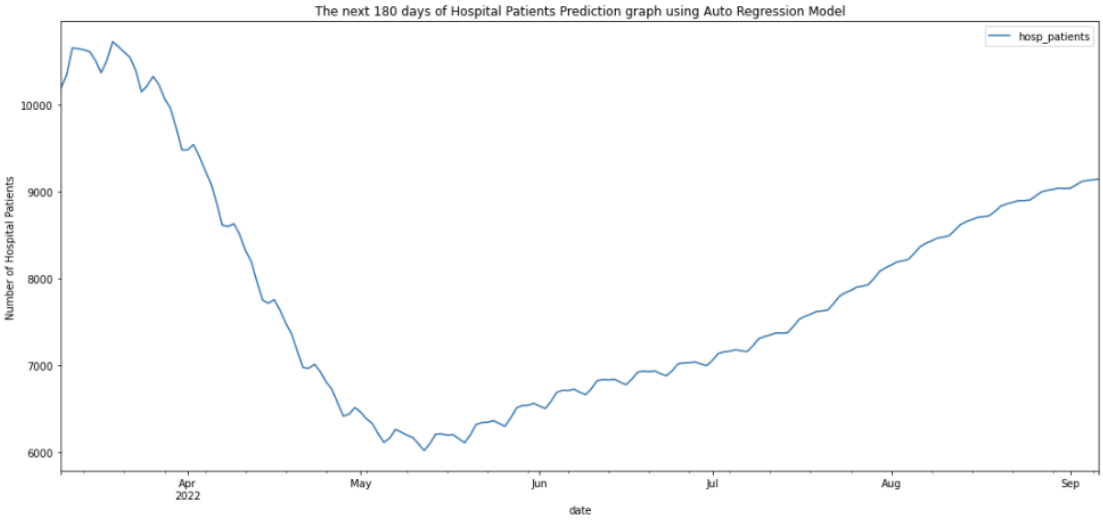


Prediction and the last 30 days of the dataset which is the test dataset of ICU Patients Dataset comparison

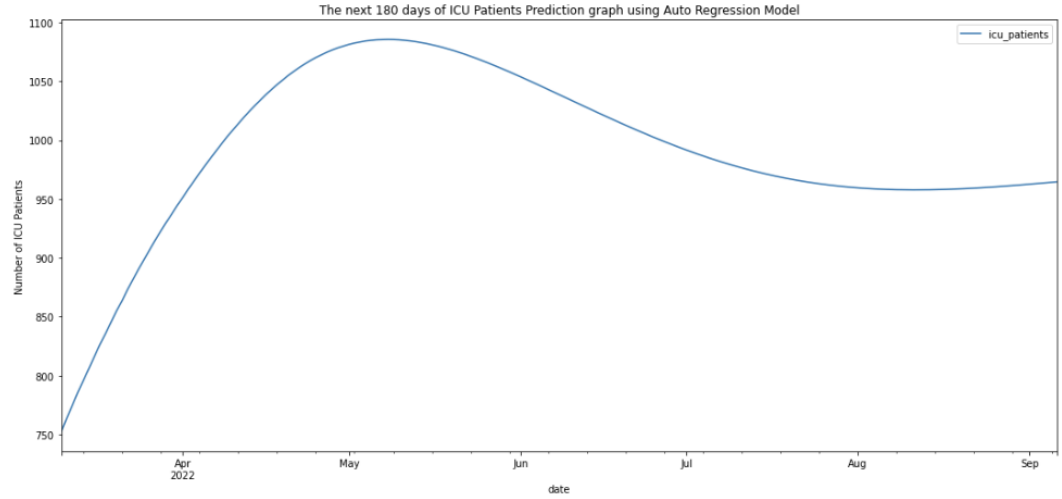


Prediction and the last 30 days of the dataset which is the test dataset of New Deaths Dataset comparison

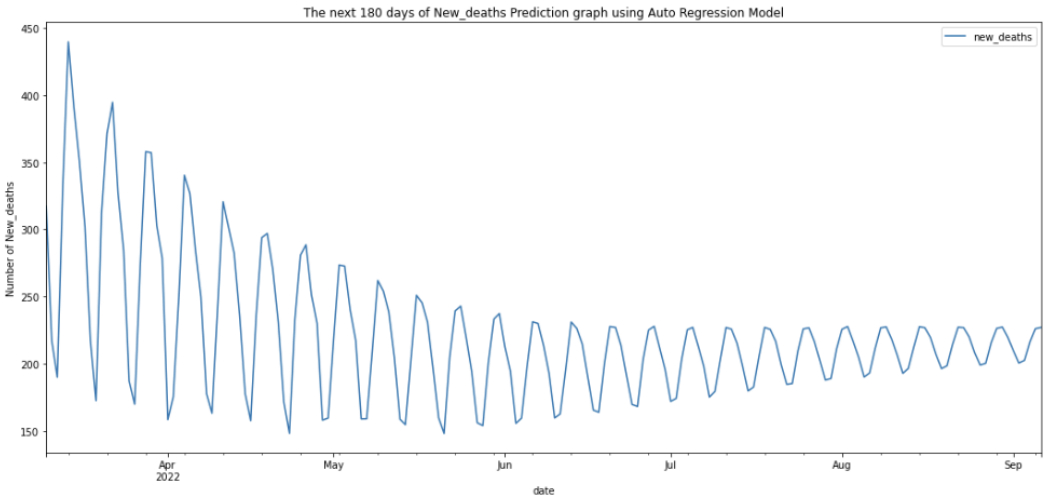
5. Make the prediction until 6th September 2022



The next 180 days(Until 6th September 2022) Prediction of Hospital Patients



The next 180 days(Until 6th September 2022) Prediction of ICU Patients



The next 180 days(Until 6th September 2022) Prediction of New Deaths

Conclusion of Auto Regression

Auto Regression is appropriate for the hospital patient datasets. It gave a high accuracy when comparing the prediction to the test dataset, and the next 180 days prediction graph is reasonable. This method is not good for every dataset. The ICU patients dataset does not fit this method. It did not perform well when comparing the prediction to the test dataset. The new deaths model using Auto Regression is not that bad in terms of predicting the trend of the new deaths in the future, but it does not perform well in predicting the exact range of numbers. This method is appropriate for some datasets only. The disadvantage point of this method is the same as ARIMA that this method is not flexible, so we cannot adjust that many parameters or tune the model.

Random Forest Regression and Linear Regression

Manage Y and X values that we are going to use for model training.

We cannot use one column as we did in ARIMA and Auto Regression model creating, so we have to select the X and Y values for the model training. For the time forecasting model, one of the valuable methods for X value preparation is to create columns that store the data of the previous days. For example, if we need to predict the hospital patients, we can make the columns that store the number of hospital patients on the previous days of each day and use them as X values. The model that we train will learn the relationship between the hospital patients each day. The picture of the example data that we used for the model creation can be seen below.

X

Y

date	hosp_patients	previous1day	previous2day	previous3day	previous4day	previous5day	previous6day	previous7day	previous8day	previous9day	...	pr
2020-07-28	1369.0	1370.0	1383.0	1401.0	1453.0	1538.0	1747.0	1634.0	1734.0	1742.0	...	
2020-07-29	1350.0	1369.0	1370.0	1383.0	1401.0	1453.0	1538.0	1747.0	1634.0	1734.0	...	
2020-07-30	1314.0	1350.0	1369.0	1370.0	1383.0	1401.0	1453.0	1538.0	1747.0	1634.0	...	
2020-07-31	1266.0	1314.0	1350.0	1369.0	1370.0	1383.0	1401.0	1453.0	1538.0	1747.0	...	
2020-08-01	1290.0	1266.0	1314.0	1350.0	1369.0	1370.0	1383.0	1401.0	1453.0	1538.0	...	
...	...	...	...	...	...	...	...	...	...	...	...	
2022-03-06	10902.0	10763.0	10772.0	10588.0	10747.0	10830.0	10872.0	10648.0	10500.0	10781.0	...	
2022-03-07	11368.0	10902.0	10763.0	10772.0	10588.0	10747.0	10830.0	10872.0	10648.0	10500.0	...	
2022-03-08	11678.0	11368.0	10902.0	10763.0	10772.0	10588.0	10747.0	10830.0	10872.0	10648.0	...	
2022-03-09	11773.0	11678.0	11368.0	10902.0	10763.0	10772.0	10588.0	10747.0	10830.0	10872.0	...	
2022-03-10	11944.0	11773.0	11678.0	11368.0	10902.0	10763.0	10772.0	10588.0	10747.0	10830.0	...	

591 rows × 180 columns

The example of data that we used for model training

After the dataset management, the last 30 days were cut out to be the test dataset. Then, we trained the model by using Random Forest Regression and Linear Regression methods.

```
In [132]: X_train,X_test,y_train,y_test=final_x[:-30],final_x[-30:],y[:-30],y[-30:]
```

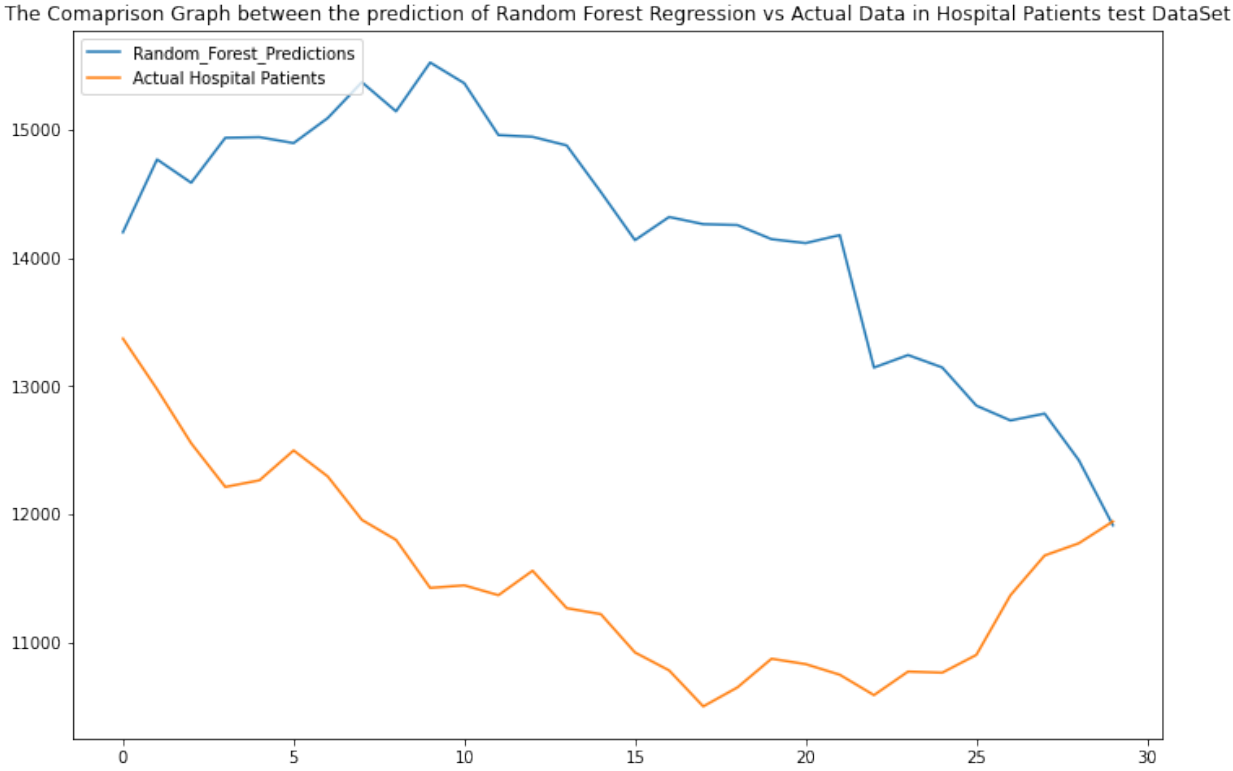
Separate the Last 30 days to be the test dataset

```
In [133]: forest_model1.fit(X_train,y_train)
          lin_model1.fit(X_train,y_train)
```

Fit the dataset to the models.

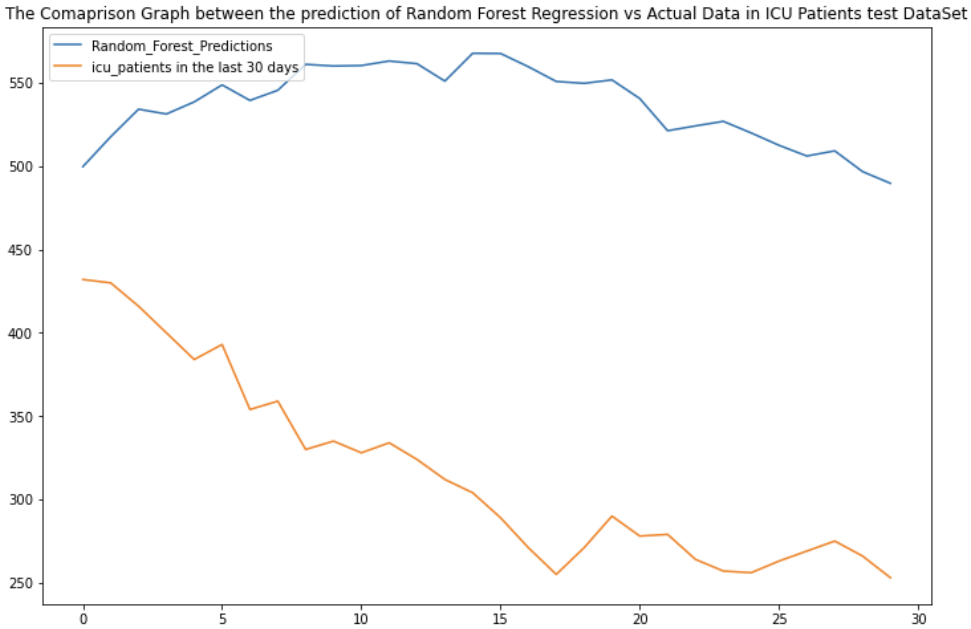
Result of the Random Forest Regression

The comparison graph between Random Forest Regression Prediction and Actual Data of Hospital Patients in the test Dataset (Last 30 Days)



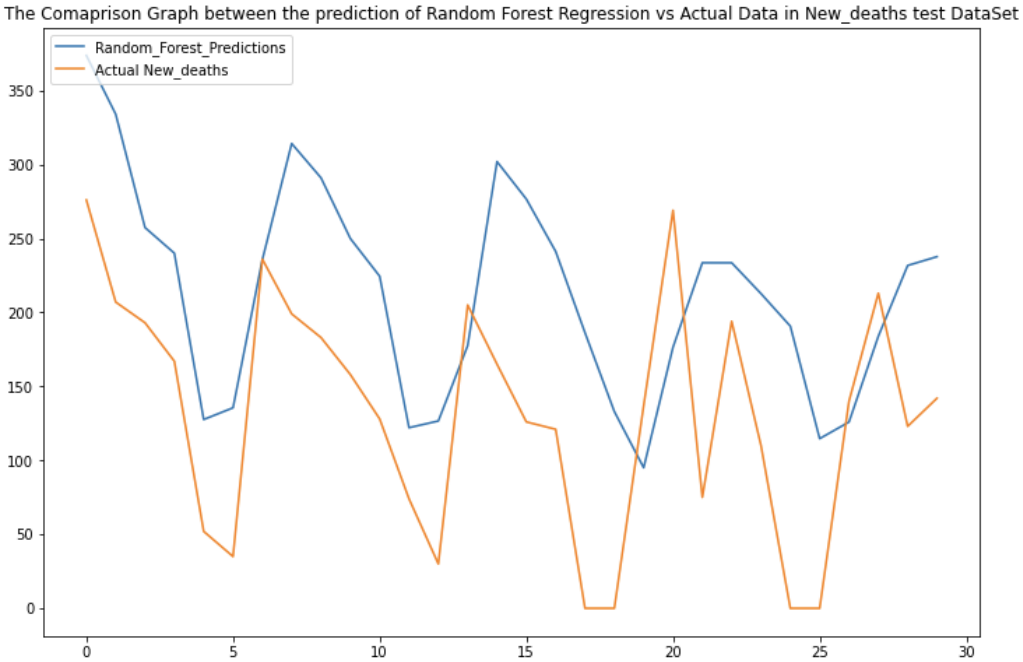
Mean Squared Error for Random Forest Model is: 2878.4373107301353

The comparison graph between Random Forest Regression Prediction and Actual Data of ICU Patients in the test Dataset (Last 30 Days)



Mean Squared Error for Random Forest Model is: 228.80401197677168

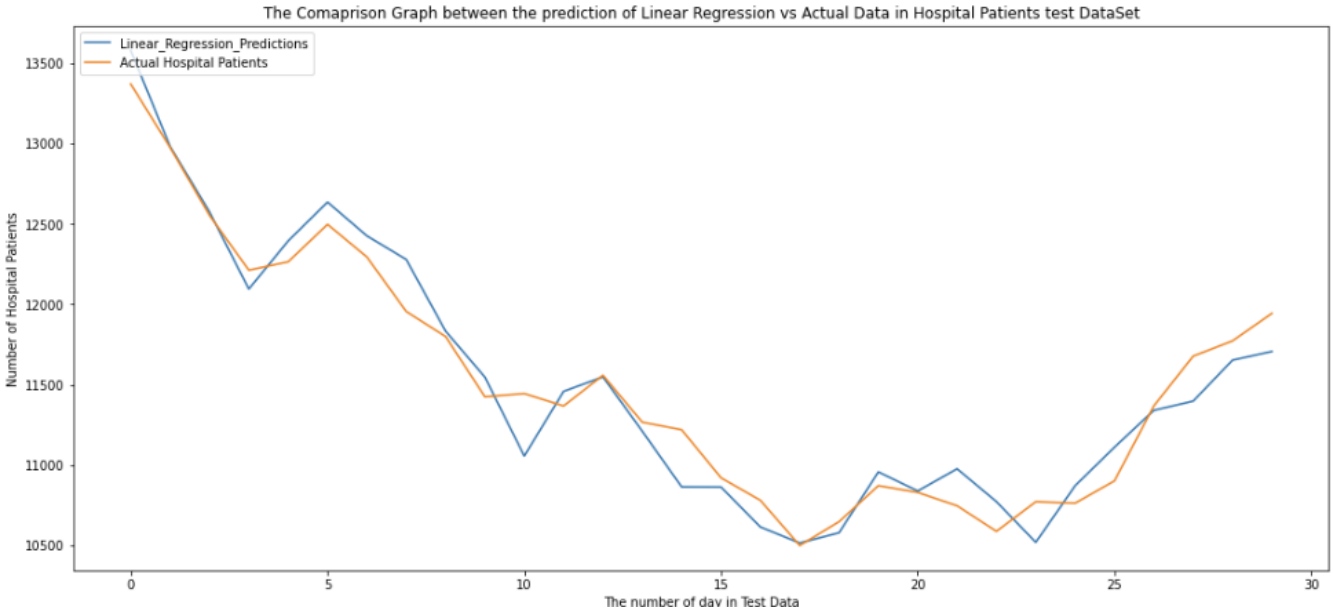
The comparison graph between Random Forest Regression Prediction and Actual Data of New Deaths in the test Dataset (Last 30 Days)



Mean Squared Error for Random Forest Model is: 105.4192943282522

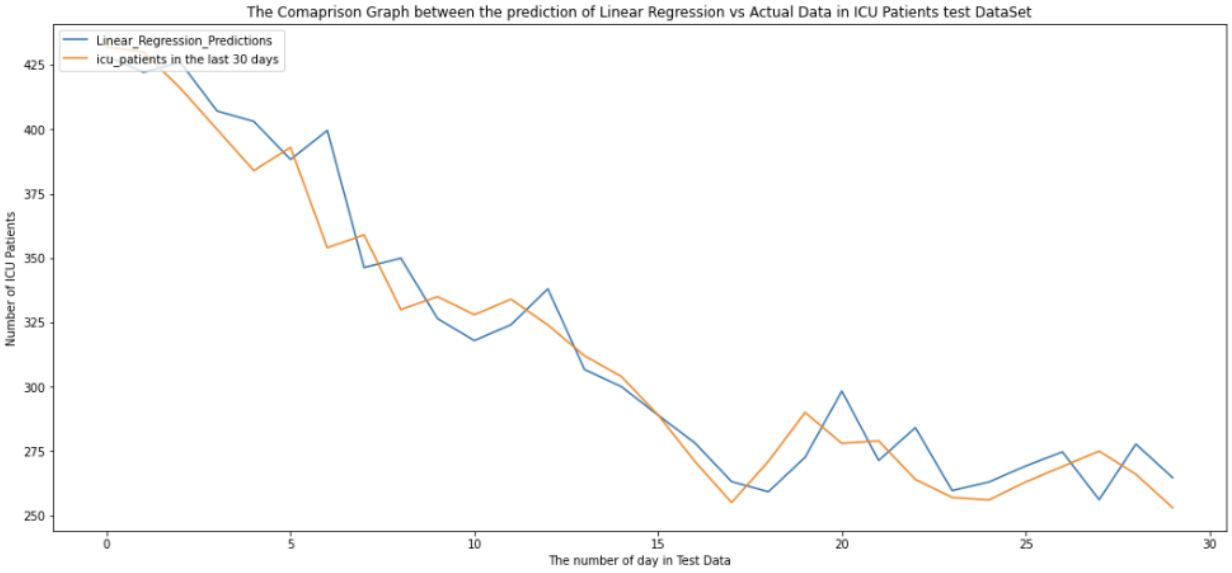
Result of the Linear Regression

The comparison graph between Linear Regression Prediction and Actual Data of Hospital Patients in the test Dataset (Last 30 Days)



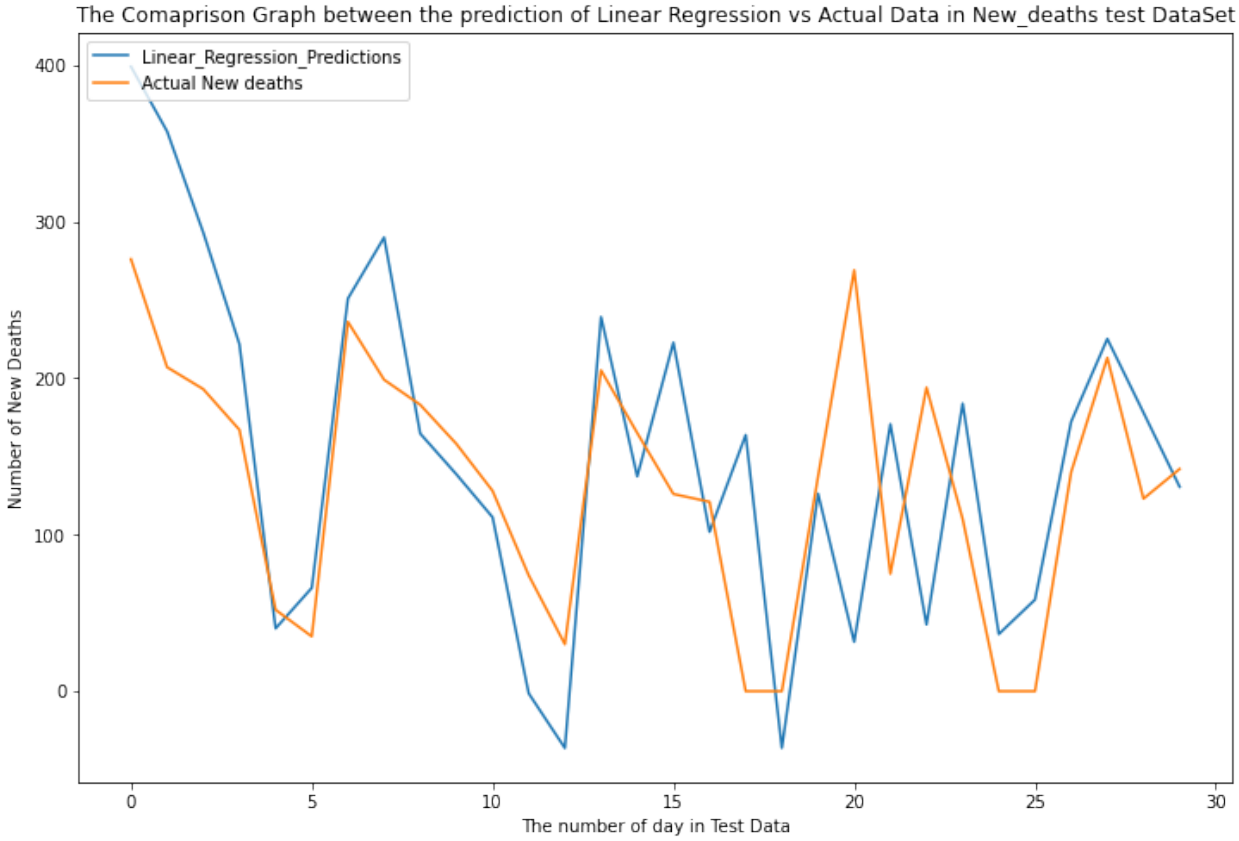
Mean Squared Error for Linear Regression Model is: 175.09629024441966

The comparison graph between Linear Regression Prediction and Actual Data of ICU Patients in the test Dataset (Last 30 Days)



Mean Squared Error for Linear Regression Model is: 14.086919501367925

The comparison graph between Linear Regression Prediction and Actual Data of New Deaths in the test Dataset (Last 30 Days)



Mean Squared Error for Linear Regression Model is: 84.53818728811906

Conclusion of Random Forest Regression and Linear Regression

The method of creating the previous days' columns and using them as X has a strongly positive result. Both Random Forest Regression and Linear Regression can use these X values to learn the relationship between the values each day and make the prediction with high accuracy. We can have more X values for the model training by using other columns to make the models perform better because it will learn about the relationship between each column. In the real world, there are many factors that affect the Y values. We will discuss this point in the discussion section. The result of the prediction from the Linear Regression model is much better than the prediction from Random Forest Regression.

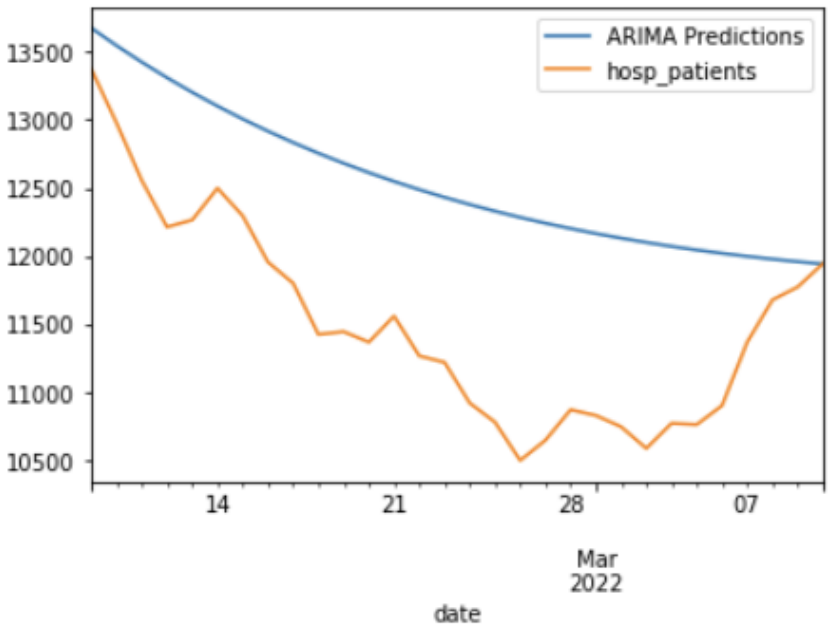
Comparison

** We tried to put every graph into one figure and write the name of each line, but it is hard to see and analyze the graph, so we decided to do the comparison by separating each graph into each figure one by one.

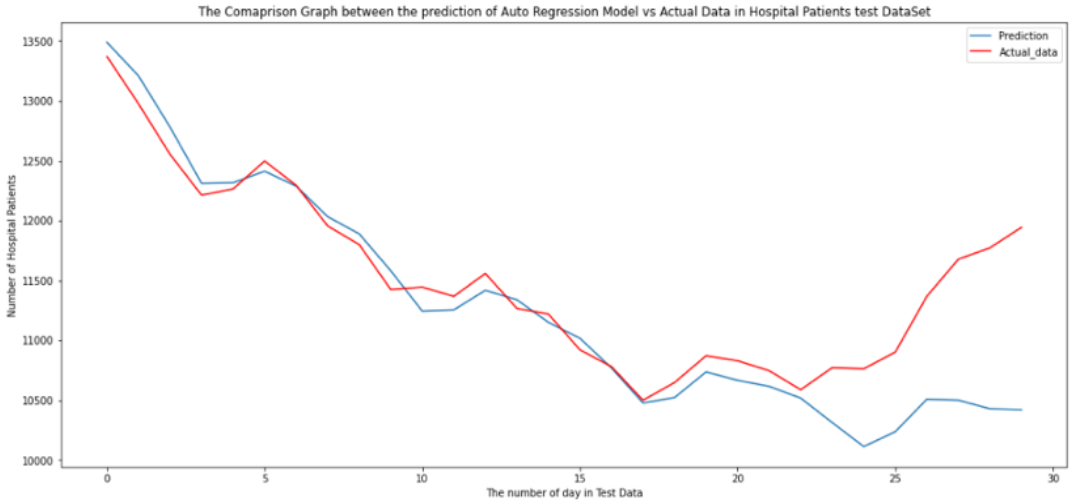
For the comparison, we used the graph that compares the prediction and the actual data in the test set (The last 30 days of the dataset).

Hospital Patients Prediction

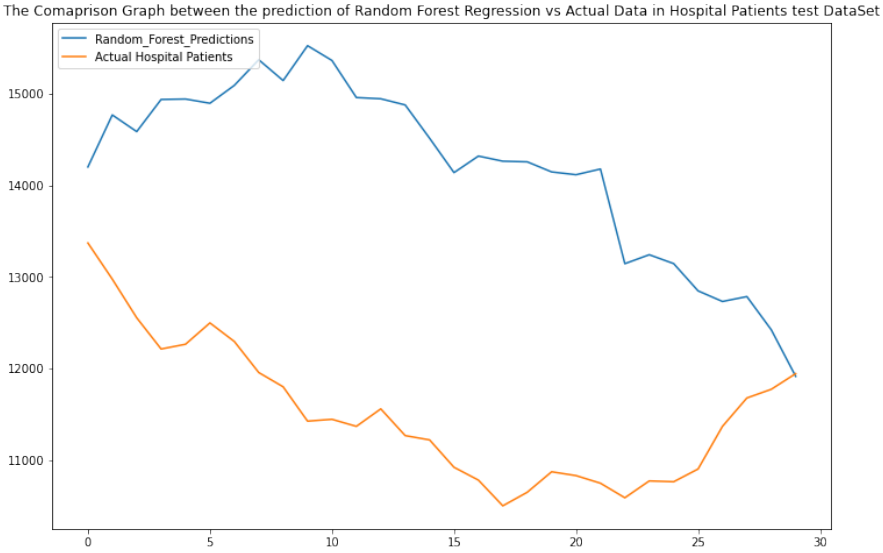
ARIMA



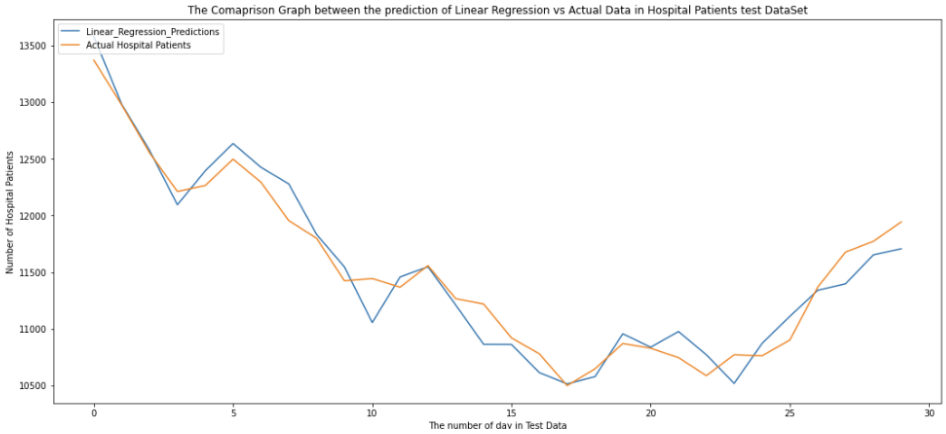
Auto Regression



Random Forest Regression



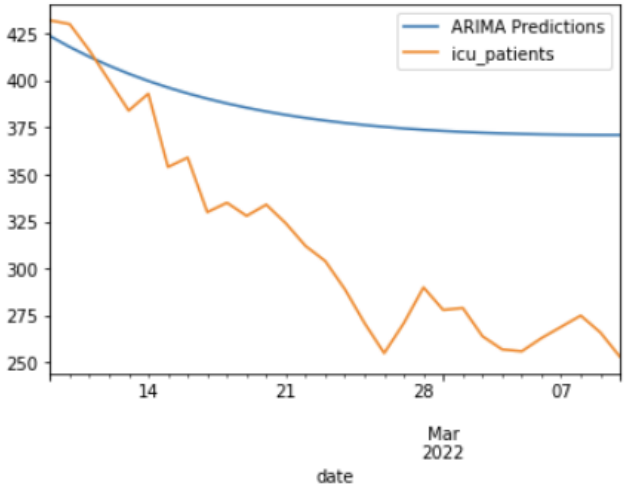
Linear Regression



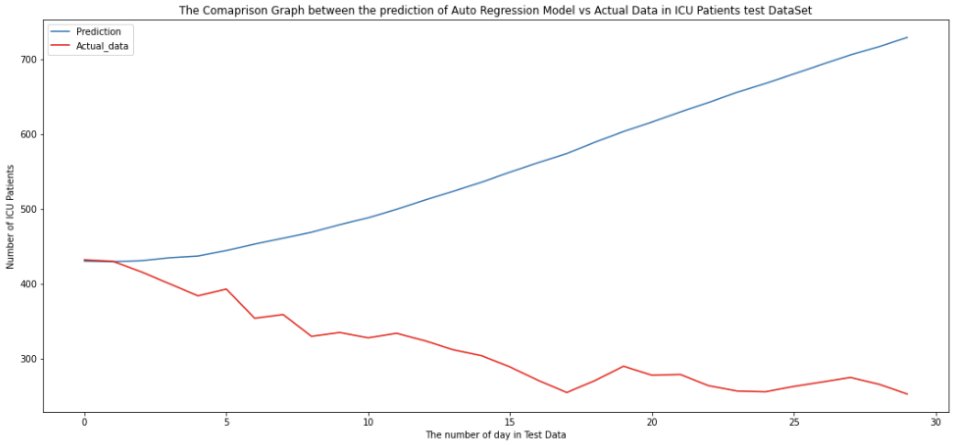
ARIMA and Random Forest Regression have the worst result. The Auto Regression seems to be the method that could be used, but at the end of the graph, it seems like the accuracy of the model is decreasing. Linear Regression is the model that gives the best accuracy compared to the result from other methods.

ICU Patients Prediction

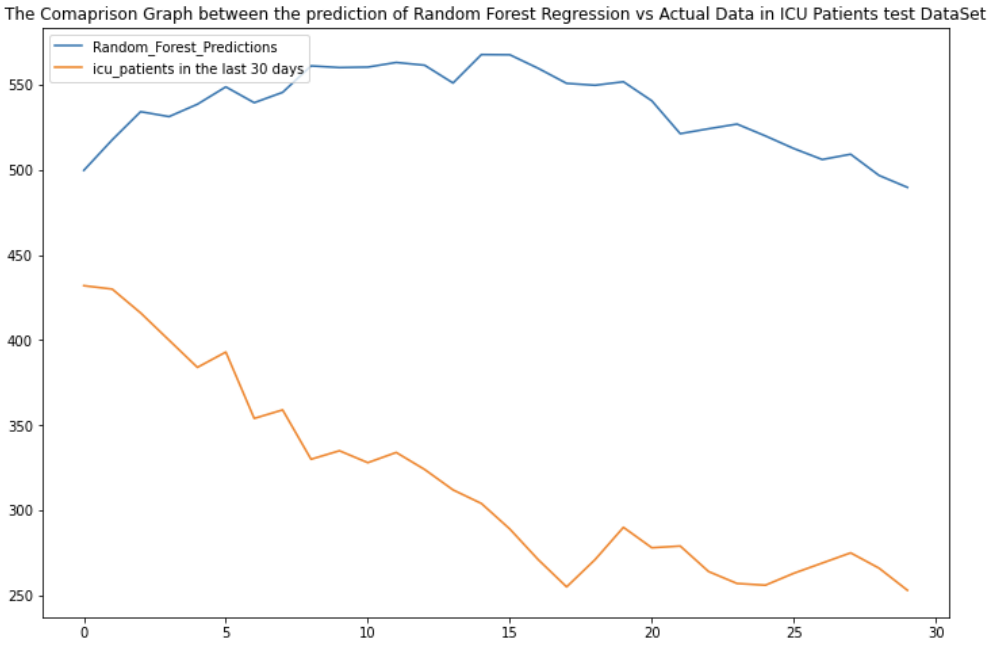
ARIMA



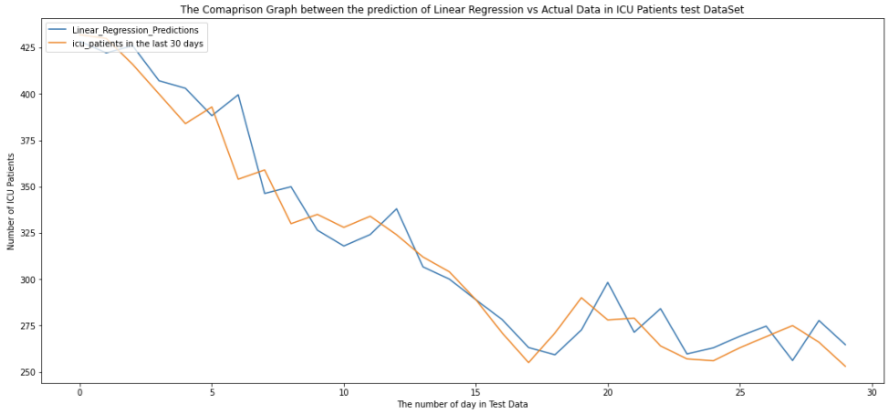
Auto Regression



Random Forest Regression



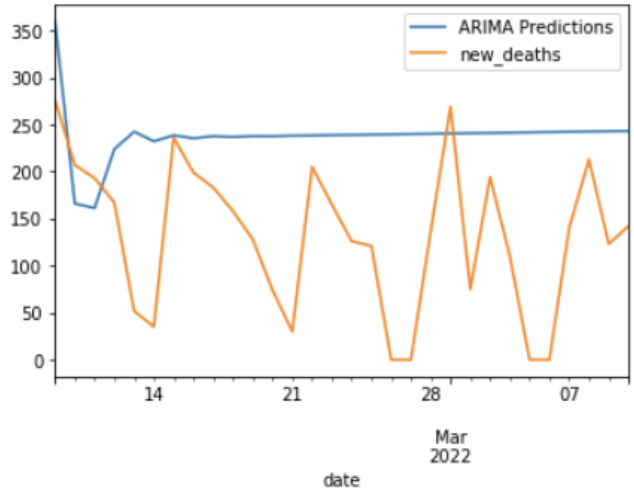
Linear Regression



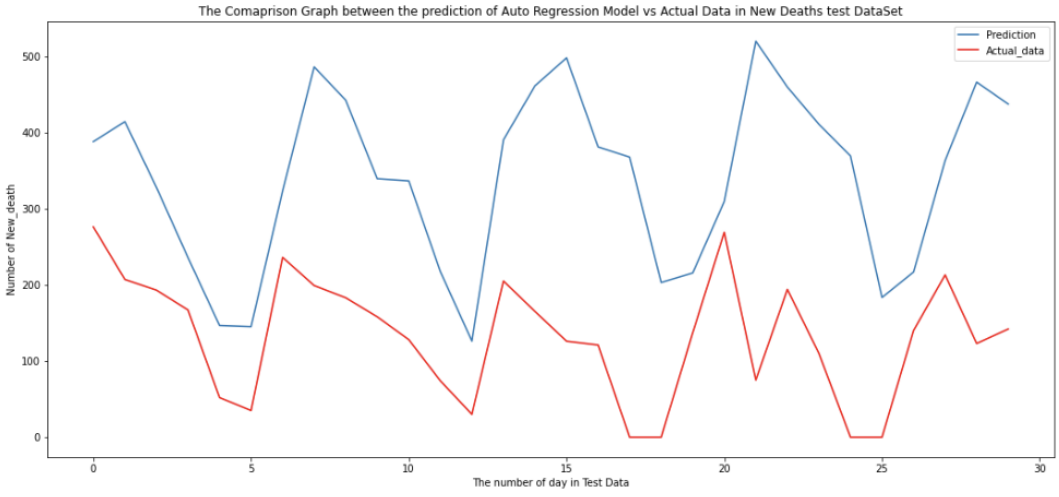
For the ICU Patients prediction model, ARIMA, Auto Regression, and Random Forest Regression perform very badly compared to the Linear Regression.

New Deaths Prediction

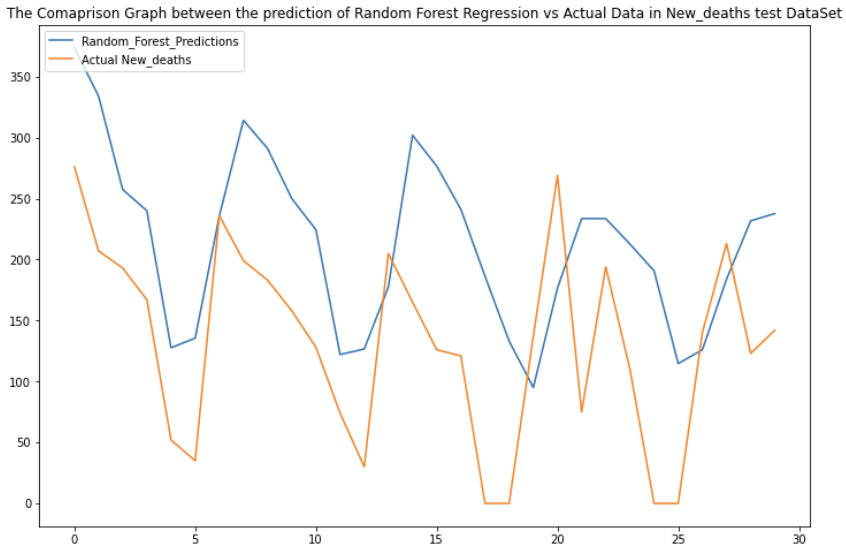
ARIMA



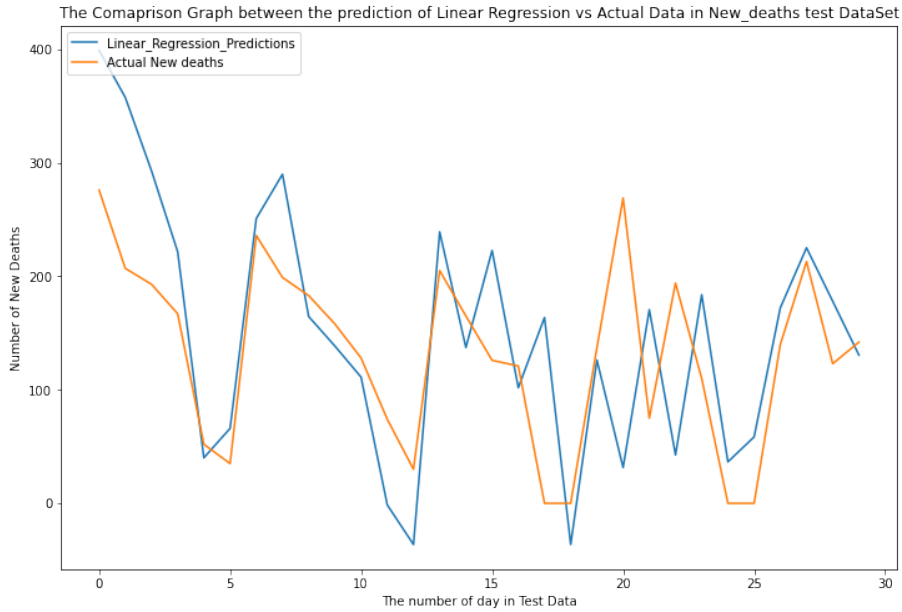
Auto Regression



Random Forest Regression



Linear Regression



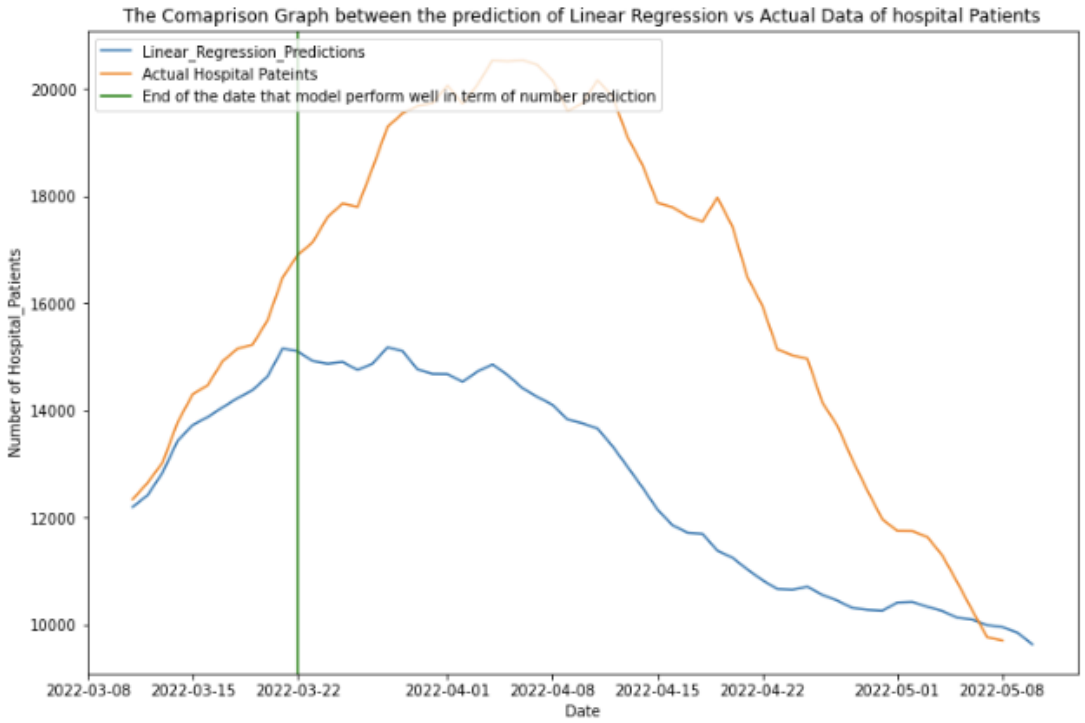
ARIMA give the bad accuracy of prediction. Auto Regression and Random Forest Regression seems to be the models that can be used to predict the trend of the dataset in the future, but they are bad in term of predicting the exact range of number. Linear Regression gives the prediction with the highest accuracy.

Evaluation With the Real Data.

We did not train data to make the models up to date because we want to make the prediction and compare the data with the actual data that we did not know.

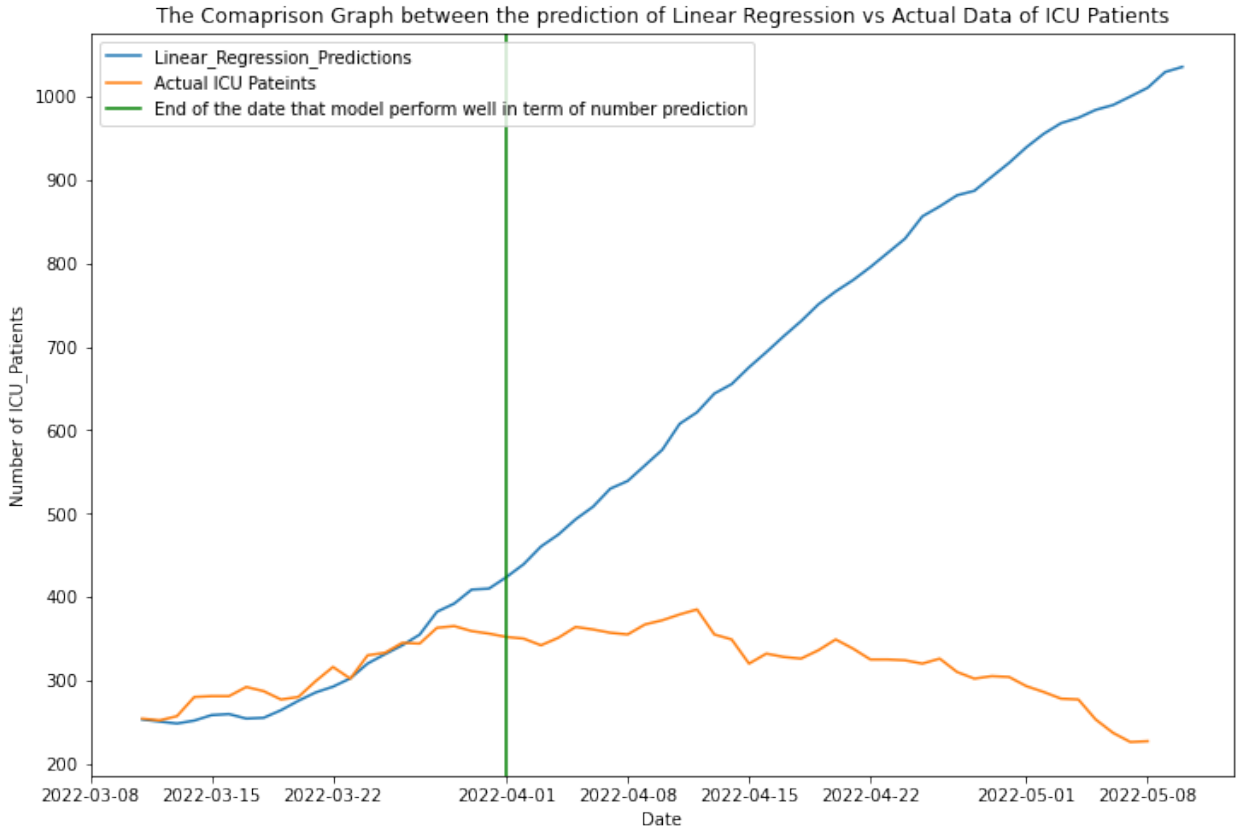
** The last day of the data we used for data training is 10th March 2022. We predicted the hospital patients, ICU patients, and new deaths until 6th September 2022. Today is 11th May 2022, so we compared the predicted data with the actual data to see the accuracy.

Hospital Patients



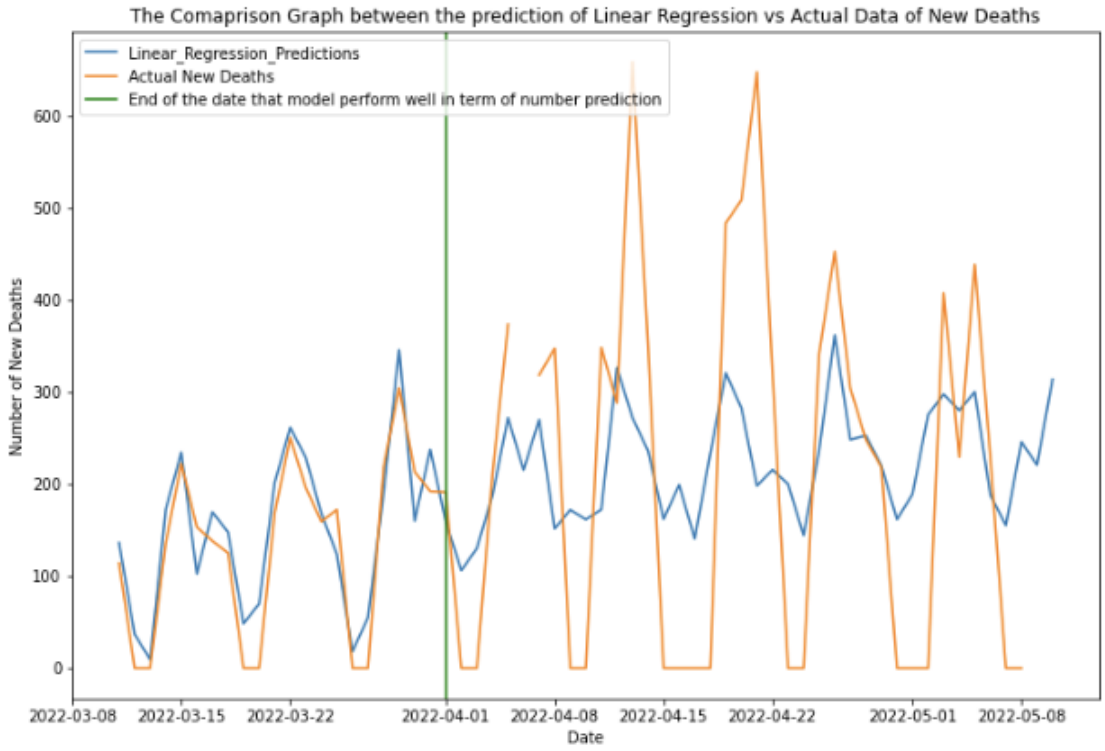
The first 12 days of the predicted data are close to the real data which is good. After 12 days, the model performs not that well in terms of exact number prediction, but the predicted dataset is still useful in terms of predicting the trend of hospital patients. The trend of data between predicted data and real data are similar to each other.

ICU Patients



For the ICU patient's prediction, the model performs well only in around 20 days. As you can see from the graph, after 20 days, the number of ICU patients and the trends of the graph are not similar to the real data.

New Deaths



There is a problem with the real data on new deaths. For some reason, new deaths in someday are zero, so it is hard to make the analysis. We found that the dataset that we used to train data also has this problem. We will talk about this problem in the discussion section. If we ignore this problem and see only the trend of the total deaths, the prediction model performs very well.

Discussion

Manage X values using the previous day method.

This method is useful and can be used in a real project, however for the version of this model, we noticed that using the previous day's value of only the column which we want to predict is not good enough. For example, if we want to predict tomorrow's value of hospital patients, creating the previous day's columns using only hospital patients' data is not good enough. The model will perform better if we create the previous day's columns from other factors and use them as X. In the real world, there are many factors that can affect the data, so the model needs more data to learn about the relationship between each column.

The picture below shows the data that we should use as X. In the version of this model, we use the previous day's columns from only one factor to do the prediction. (We use previous day columns of hospital Patients to predict hospital patients, and use previous day columns of ICU Patients to predict hospital patients.) It is not that bad, but the model can perform better if we add data from previous day columns from other factors to X.

icu_patients	previous1day	previous2day	previous3day	previous4day	previous5day	previous6day	previous7day	previous8day	previous9day	...
93.000000	97.000000	92.000000	93.000000	104.000000	111.000000	110.000000	137.000000	129.000000	131.000000	...
87.000000	93.000000	97.000000	92.000000	93.000000	104.000000	111.000000	110.000000	137.000000	129.000000	...
84.000000	87.000000	93.000000	97.000000	92.000000	93.000000	104.000000	111.000000	110.000000	137.000000	...
86.000000	84.000000	87.000000	93.000000	97.000000	92.000000	93.000000	104.000000	111.000000	110.000000	...
92.000000	86.000000	84.000000	87.000000	93.000000	97.000000	92.000000	93.000000	104.000000	111.000000	...
...	...	...	...	...	...	...	...	...	...	...
1221.917005	1229.653318	1239.107437	1247.618639	1255.311134	1259.074721	1264.139394	1267.859289	1274.100288	1280.947904	...
1215.637900	1221.917005	1229.653318	1239.107437	1247.618639	1255.311134	1259.074721	1264.139394	1267.859289	1274.100288	...
1209.541216	1215.637900	1221.917005	1229.653318	1239.107437	1247.618639	1255.311134	1259.074721	1264.139394	1267.859289	...
1205.018877	1209.541216	1215.637900	1221.917005	1229.653318	1239.107437	1247.618639	1255.311134	1259.074721	1264.139394	...
1196.652489	1205.018877	1209.541216	1215.637900	1221.917005	1229.653318	1239.107437	1247.618639	1255.311134	1259.074721	...

hosp_patients	previous1day	previous2day	previous3day	previous4day	previous5day	previous6day	previous7day	previous8day	previous9day	...	pr
1369.0	1370.0	1383.0	1401.0	1453.0	1538.0	1747.0	1634.0	1734.0	1742.0	...	
1350.0	1369.0	1370.0	1383.0	1401.0	1453.0	1538.0	1747.0	1634.0	1734.0	...	
1314.0	1350.0	1369.0	1370.0	1383.0	1401.0	1453.0	1538.0	1747.0	1634.0	...	
1266.0	1314.0	1350.0	1369.0	1370.0	1383.0	1401.0	1453.0	1538.0	1747.0	...	
1290.0	1266.0	1314.0	1350.0	1369.0	1370.0	1383.0	1401.0	1453.0	1538.0	...	
...	...	...	...	...	...	...	...	...	...	...	
10902.0	10763.0	10772.0	10588.0	10747.0	10830.0	10872.0	10648.0	10500.0	10781.0	...	
11368.0	10902.0	10763.0	10772.0	10588.0	10747.0	10830.0	10872.0	10648.0	10500.0	...	
11678.0	11368.0	10902.0	10763.0	10772.0	10588.0	10747.0	10830.0	10872.0	10648.0	...	
11773.0	11678.0	11368.0	10902.0	10763.0	10772.0	10588.0	10747.0	10830.0	10872.0	...	
11944.0	11773.0	11678.0	11368.0	10902.0	10763.0	10772.0	10588.0	10747.0	10830.0	...	

new_deaths	previous1day	previous2day	previous3day	previous4day	previous5day	previous6day	previous7day	previous8day	previous9day	...	previo
21.0	3.0	8.0	15.0	32.0	9.0	17.0	25.0	10.0	11.0	...	
34.0	21.0	3.0	8.0	15.0	32.0	9.0	17.0	25.0	10.0	...	
0.0	34.0	21.0	3.0	8.0	15.0	32.0	9.0	17.0	25.0	...	
20.0	0.0	34.0	21.0	3.0	8.0	15.0	32.0	9.0	17.0	...	
13.0	20.0	0.0	34.0	21.0	3.0	8.0	15.0	32.0	9.0	...	
...	...	...	...	...	...	...	...	...	...	...	
0.0	0.0	110.0	194.0	75.0	269.0	137.0	0.0	0.0	121.0	...	
140.0	0.0	0.0	110.0	194.0	75.0	269.0	137.0	0.0	0.0	...	
213.0	140.0	0.0	0.0	110.0	194.0	75.0	269.0	137.0	0.0	...	
123.0	213.0	140.0	0.0	0.0	110.0	194.0	75.0	269.0	137.0	...	
142.0	123.0	213.0	140.0	0.0	0.0	110.0	194.0	75.0	269.0	...	

rows × 180 columns

If you need to create the model to predict hospital patients, use Previous day columns of hospital patients, ICU patients, and new deaths as X. Do not use only previous days columns from hospital patients.

New Deaths Dataset Problem

For some reason, there are many rows in the new deaths column that are filled in with zero which is not good for model creating. It is hard to do the prediction analysis as well as evaluate the model accuracy. If you need to create a model to predict the new deaths in the future, make sure that you have already dealt with this problem. If the objective of the model creation is to create a model that can show the trend of new deaths, you can ignore this problem. It is better not to create the model using new death columns of England in the Covid-19 dataset.

Conclusion

There are many methods to create a time forecasting machine learning model. There is no best method. The best way to get a good model is to try every method and do a comparison. In this project, we found that using Linear Regression as the method to create the time forecasting model for hospital patients, ICU patients, and new deaths is one of the best methods so far. From the comparison between the prediction dataset (Prediction dataset by using the model that trains by the dataset that has last data of 10th March 2022) and real data (The actual real data that have data until 10th May 2022), hospital patient and new deaths columns in the prediction dataset have the trend similar to the real data, but ICU patient column does not have a similar trend to the real dataset (The result and more information can be read in the Evaluation with the Real Data section). There are many things we noticed that can be done to improve the model. We collected them and noted them down in the discussion section. In the future, we will try to improve the model in order to have better accuracy and be useful in the real world.