Data Analytic ECS784P

Data Analysis of Novel Coronavirus Data in Worldwide

Table of individual contribution by each member of the group, based on the subjective opinion:

Student	Effort
Shengxi Cui	25%
Ziang Xie	25%
Dian Ding	25%
Jiaxue Tian	25%

Name: Dian Ding
Number: 190397821
Group size: 4

Total number of words(including references): 4374

Table of Contents

Part 1 Data modeling and analysis

1.Introduction and Background

Coronavirus is a family of viruses that can cause illness, which can vary from common cold and cough to sometimes more severe disease.SARS-CoV-2 (n-coronavirus) is the new virus of the coronavirus family, which first discovered in 2019, which has not been identified in humans before. Which later declared as Pandemic by WHO due to high rate spreads throughout the world. Currently (on the date 13 May 2020), this leads to a total of 290k Deaths across the globe, including 156k deaths alone in Europe.

Pandemic is spreading all over the world; it becomes more important to understand about this spread. This project is an effort to analyze the cumulative data of confirmed deaths and recovered cases over time. In this project, the main focus is to analyze the spread trend of this virus all over the world and provide reliable life advice and policy adjustments based on data analysis.

2.Project Aims

With the proposal of various machine learning algorithms, the ability of computers to make accurate predictions is becoming more durable and more reliable, and the impact of data analysis in various fields is becoming more and more profound. In ECS784P, we have mainly studied algorithms related to data analysis and the mathematical principles behind them. As the coronavirus spread throughout the world, we decided to use what we learned to make a predictive analysis of the epidemic. A few countries' policies with significant differences have been carried out by which has been picked up and some data such as coronavirus cases, mortality rates, and recovery rates from then grabbed. After operating (cleaning, reshaping, etc.) to the data, four models, including linear regression, Polynomial regression, and long - short - term memory - networks have had 2 to establish the prediction. Hopefully, the result given by the previous data -based one to algorithms could give some constructive suggestions on how do we carry out policy.

3.Literature Review

In response to global dispersion of severe acute respiratory syndrome–coronavirus 2 (SARS-CoV-2), quarantine measures have been implemented around the world., the analysis of epidemic data is critical to epidemic control, economic recovery and policy formulation, as well as to correct misperceptions.

1) In the exploration of the non-pharmaceutical factors and interventions of Covid-19, Flaxman et al. (2020) made a thorough analysis of the interventions adopted by European countries by building a death model and infection model, which inspired us both on the prediction modeling and the drawing of a causal graph.
2) Wells et al. (2020) analyzed international travel and border control measures on the

global spread. They made predictions based on the travel history to China, which inspired us on the variable and model choice when making an estimation.

3) Chinazzi (2019) also proved that the travel quarantine introduced in Wuhan on 23 January 2020 only delayed epidemic progression by 3 to 5 days within China is wrong.
4) Švaňa (2018) explained Support Vector Machine (SVM) classifier implementations in Python, which provided us a deep understanding of the SVM algorithm.
5) Furthermore, Géron (2019) explained detailed machine learning libraries, including Scikit-learn, Keras, and TensorFlow, which provided us a hand-on-hand guide on the modeling work.
6) For data analytics tools, McKinney (2019) made a comprehensive introduction to the Python library, which helped us have a better understanding of choosing and applying the library to our work.
7) Rossant (2013) provided a detailed data visualization by Python guide that helped us with the data visualization work.
8) For the BN structure learning, Constantinou (2020) introduced a hybrid Bayesian Network algorithm that involves both constraint-based learning and scored-based learning called SaiyanH. He uses SaiyanH to produce a scored DAG that explained the causal relationships among nodes. The Bayesian analysis in this report is based on the SaiyanH algorithm.
9) Shuzhi Li (2010) also contributed a deep understanding of the Bayesian network structure score function states by stating that the Bic score function was applied to structure learning of the Asia network.

4. Data collection

The data utilized for the analysis was collected from GitHub. Epidemic data is provided by Johns Hopkins University. JHU is currently using a GitHub repo to store its data instead of a web app. I will need to update my scraping methodology. Apologies for not being able to provide the most up-to-date coronavirus dataset. Alternatively, if you are very excited about the dataset, JHU already updates their dataset daily on this link:

https://github.com/CSSEGISandData/COVID-19/blob/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_confirmed_global.csv

The datasets obtained were:

- time_series_covid19_recovered_global.csv
- time_series_covid19_deaths_global.csv
- time_series_covid19_confirmed_global.csv
- csse_covid_19_daily_reports
- population_by_country_2020.csv

We used the method of getting data from the website in real-time to get some data：

```
confirmed_df = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/C
deaths_df = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVI
recoveries_df = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/
```

For ease of operation, we consolidated the data into a single file：

country_wise_pop_weekly_latest.csv

```
In [225]: #The structure of country_wise_pop_weekly_latest.csv
          country_wise_pop_weekly_latest.head(10)
```

Out[225]:

try/Region	Confirmed	Deaths	Recovered	Active	New cases	Deaths / 100 Cases	Recovered / 100 Cases	Deaths / 100 Recovered	Population	Yearly Change	Net Change	Density (P/Km²)	Land Area (Km²)	Cases / Million People	Confirme last wee
Afghanistan	2171	64	260	1847	232	2.95	11.98	24.62	38742911.0	2.33 %	886592.0	60.0	652860.0	56.0	127
Albania	773	31	470	272	7	4.01	60.80	6.60	2878420.0	-0.11 %	-3120.0	105.0	27400.0	269.0	66
Algeria	4006	450	1779	1777	158	11.23	44.41	25.30	43685618.0	1.85 %	797990.0	18.0	2381740.0	92.0	300
Andorra	745	42	468	235	2	5.64	62.82	8.97	77240.0	0.16 %	123.0	164.0	470.0	9645.0	72
Angola	27	2	7	18	0	7.41	25.93	28.57	32644783.0	3.27 %	1040977.0	26.0	1246700.0	1.0	2
Antigua and Barbuda	24	3	11	10	0	12.50	45.83	27.27	97764.0	0.84 %	811.0	223.0	440.0	245.0	2
Argentina	4428	218	1256	2954	143	4.92	28.36	17.36	45111229.0	0.93 %	415097.0	17.0	2736690.0	98.0	343
Armenia	2066	32	929	1105	134	1.55	44.97	3.44	2962137.0	0.19 %	5512.0	104.0	28470.0	697.0	152
Australia	6766	93	5742	931	14	1.37	84.87	1.62	25439164.0	1.18 %	296686.0	3.0	7682300.0	266.0	666
Austria	15452	584	12907	1961	50	3.78	83.53	4.52	8996022.0	0.57 %	51296.0	109.0	82409.0	1718.0	1500

Figure 1: The structure of country_wise_pop_weekly_latest.csv as a dataframe

5.Data Representation

The Python is widely adopted as the dominant programming language in the data science area for the multiple programming paradigms of it. It is usually involved in imperative and object-oriented functional programming. It has a comprehensive and extensive standard library that has automatic memory management and dynamic features.

Specifically, we made use of the following libraries to do the primary analysis :

- **Pandas**: a prominent open-source data manipulation library. It simplifies irretrievably of data from external sources and provides high performance, easy-to-use data structures, and data analysis tools. For this project, the main Pandas functionalism utilized is the Dataframe and Series objects.
- **NumPy:** a fundamental package for scientific computing. It provides support for large multidimensional arrays with high-level efficient math functions for operations on these arrays. In particular, many machine learning algorithms can be expressed as a sequence of operations on arrays using NumPy.
- **Matplotlib:** a basic Python package with a variety of functions for data plotting and visualization such as plots, maps, charts, etc. It is a plotting library for the Python programming language and its numerical mathematics extension NumPy.
- **Scikit-learn:** Scikit-learn is a free software machine learning library for the Python programming language.[3] It features various classification, regression and clustering algorithms, including support vector machines, random forests, gradient boosting, k-means, and DBSCAN. It is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.
- **Keras:** Keras is an open-source neural network library written in Python. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible.

6.Data Processing

In the real world, data is often incomplete (missing some property value of interest), inconsistent (containing code or name differences), and susceptible to noise (errors or outliers).Because databases are too large and data sets often come from multiple

heterogeneous data sources, low-quality data leads to low-quality mining results.To perform the analysis effectively, various data cleaning techniques were carried out.

This part is mainly divided into the following steps:

- **Importing datasets**

DataFrame is a 2-dimensional labeled data structure with columns of potentially different types. Hare we use .read_csv() function to import data as dataframe.

```
#Creating the DataFrames
confirmed_df = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/
deaths_df = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/ti
recoveries_df = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_serie
population_data=pd.read_csv('C:/Users/1/Desktop/数据集/DATA/population_by_country_2020.csv')
full_table = pd.read_csv('C:/Users/1/Desktop/数据集/DATA/full_table_clean.csv',parse_dates=['Date'])
country_wise_pop_weekly_latest = pd.read_csv('C:/Users/1/Desktop/数据集/DATA/country_wise_pop_weekly_latest.csv')
latest_data = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_daily_report
```

Figure 2 Data import

- **Check data**

First, we take a quick look at the format of the data by using ***.head()*** function.

Using ***.Date.value_counts().sort_index()***,function, we can get the total period of data.

```
a = full_table.Date.value_counts().sort_index()
print('days covered:',len(a))
print('The first date is:',a.index[0])
print('The last date is:',a.index[-1])
```

```
days covered: 100
The first date is: 2020-01-22 00:00:00
The last date is: 2020-04-30 00:00:00
```

Figure 3

Checking for missing data is also a critical step,***.isnull().sum()*** will return the total number of empty value.

- **Adding columns**

We sorted out the obtained data and added some new features (including 'Cases/Million People,' '1-week change', etc.). This makes the data attributes more comprehensive and readable.

```
full_table['Active'] = full_table['Confirmed'] - full_table['Deaths'] - full_table['Recovered']
```

Figure 4 Add columns

- **Removing Duplicates**

In the process of massive data collection, it is inevitable that there will be data duplication, the function ***.duplicated()*** was used to remove duplicate entries.

- **Dealing with Missing Values**

We check for missing values using***. IsNull()*** function. Missing values are problematic in that there are variations in the ways and reasons data are missing that require different approaches to handle them accordingly.

- **Checking data type**

When some mathematical operations are used, the format of the data affects the result of the operation. We use ***.astype()*** function to check data type.

- **data combination**

A ***.groupby()*** operation involves some combination of splitting the object, applying a function, and combining the results.

- **merged data**

After processing the data, we use ***the merge()*** function to merge the tables.

- **Save the new data table.**

Finally, use ***.to_csv()*** function to save our new dataframe as a .csv file.

7.Data Exploration and visualization

1. World map

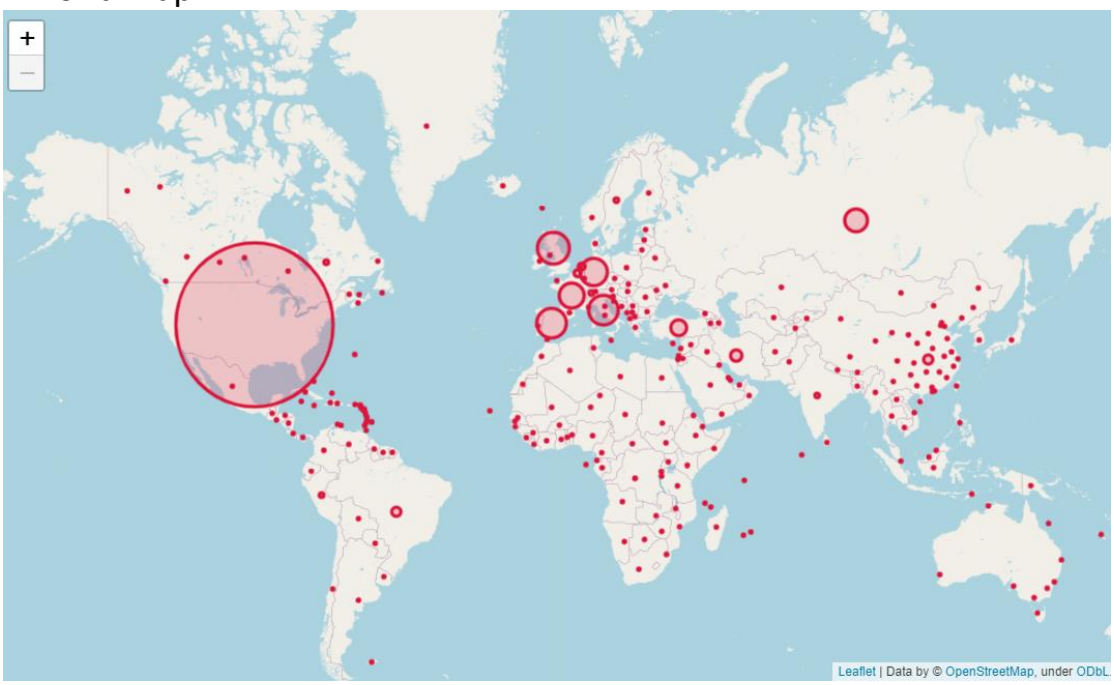


Figure 5 Geographical visualization of Coronavirus Spread

We called the ***folium.Map()*** function to display the world Map. By looking at the size of the red dots, we can get a sense of what is happening in the region. You can see that from the diagram, the epidemic has spread around the world. At present, the epidemic is developing rapidly in Europe and North America, which have become the hardest-hit areas.

2. Global epidemic trends

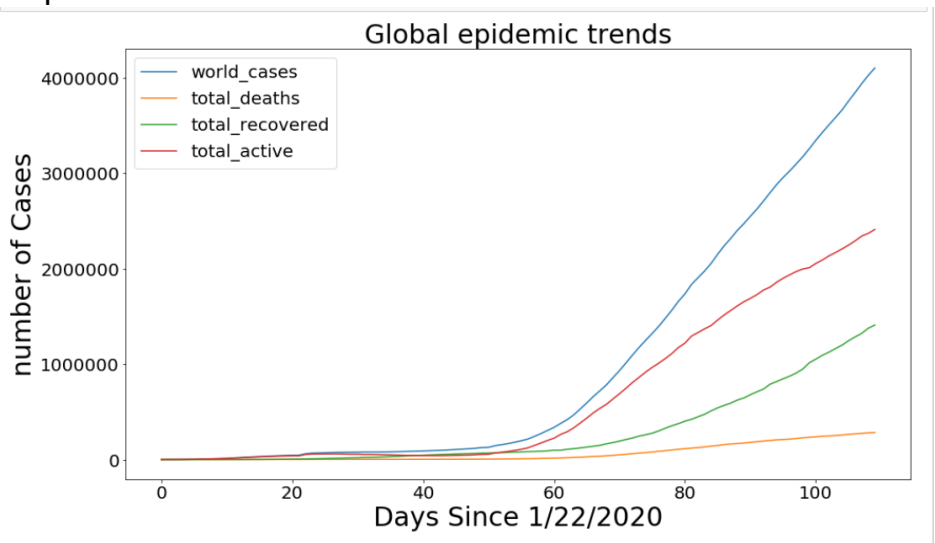


Figure 6 Global epidemic trends Visualization

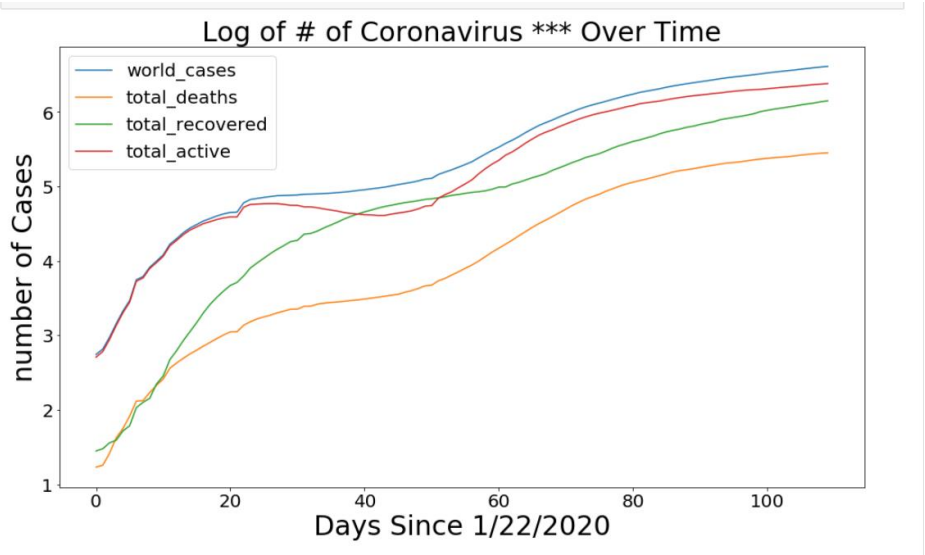


Figure 7 Log Visualization

Integrate the cumulative number of confirmed cases、deaths、recovered and active cases in the world into one chart, and the trend of the epidemic was obtained by using a logarithmic function. Start with observation day (1/22/2020), As we can see from the figure, the number of confirmed cases is already high, and the number of deaths is rising.

3. Mortality rate and Recovery rate

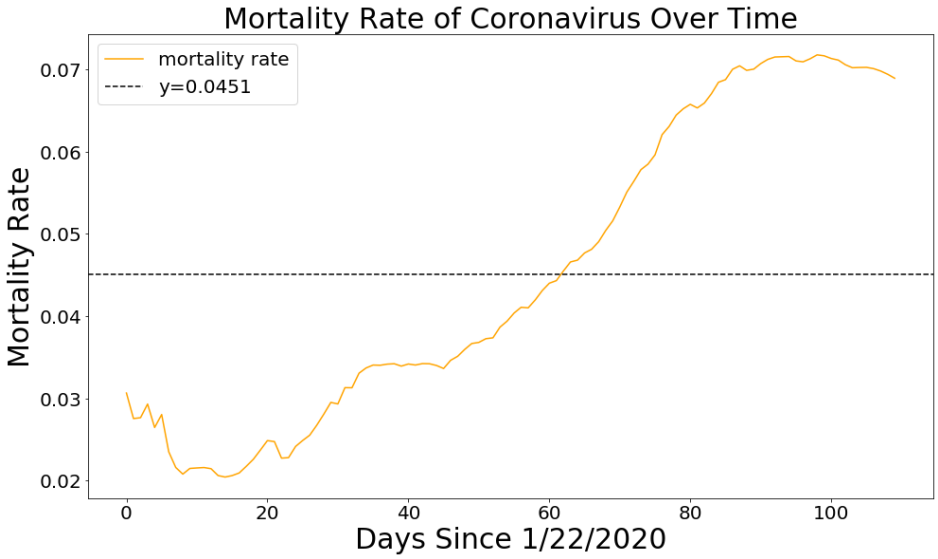


Figure 8 Visualization of Mortality Rate of Covid-19

As is shown in the figure, the mortality rate of coronavirus has been decreasing since 1/22/2020 for about 20 days. However, it has increased a significant amount in the next 70 days, and the cause of this phenomenon may be the time lag of the observation. Mortality has decreased by a few amounts since April given advantage of the policy, such as isolation and more medical facilities carried out by governments.

4. Epidemic situation in different countries

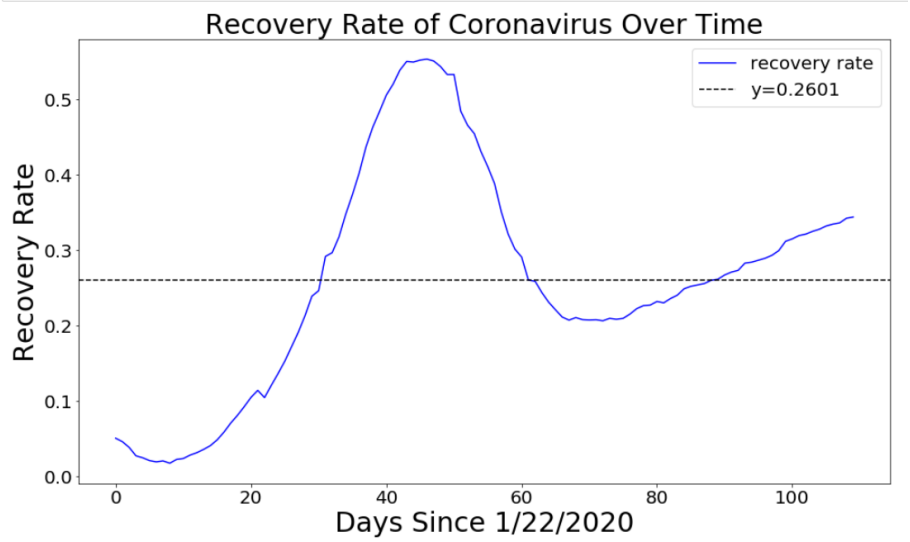


Figure 9 Visualization of Recovery Rate

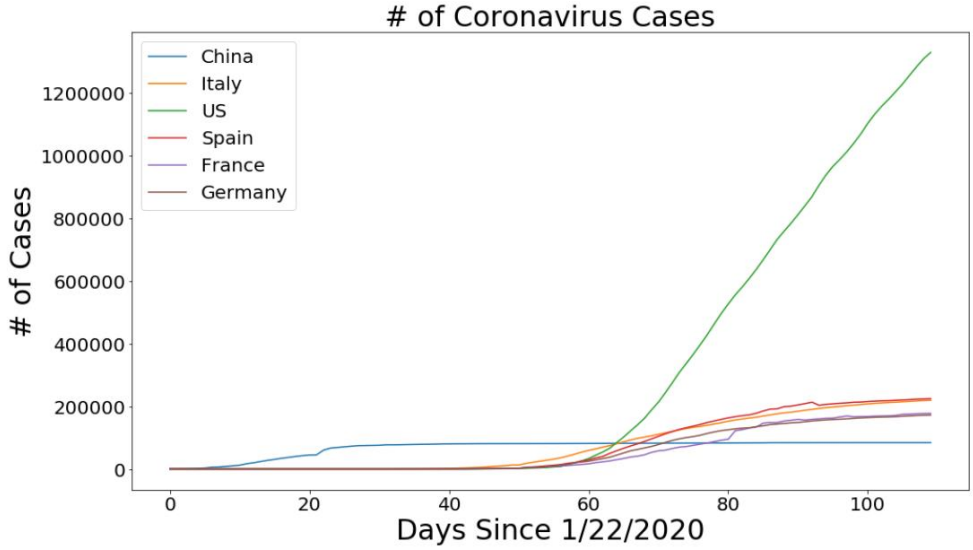


Figure 10 Visualization of confirmed cases

The figure combined the recovery rate of the coronavirus cases shown above. There is a stagnation point of recovery rate at the beginning of March, and coronavirus cases have increased by a significant amount since April, at the same time that mortality began to increase. Underestimation and neglect of the spreading ability of such a virus might be the leading cause.

We encourage you to check the appendices section for more interesting visualizations.

8.Modeling

Linear model

- **SVM**

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression, and outlier detection.

The advantages of support vector machines are:

Effective in high dimensional spaces. 2.Still useful in cases where the number of dimensions is higher than the number of samples. 3. It uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.

The algorithm is divided into the following steps:

1. split the data into train and test sets.
2. Use the SVM model to fit the data
3. Plot for Cross-Validation

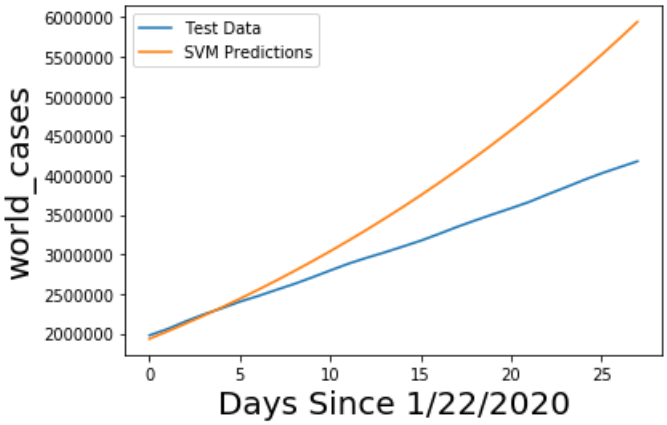


Figure 12 SVM prediction

4. Output forecast data

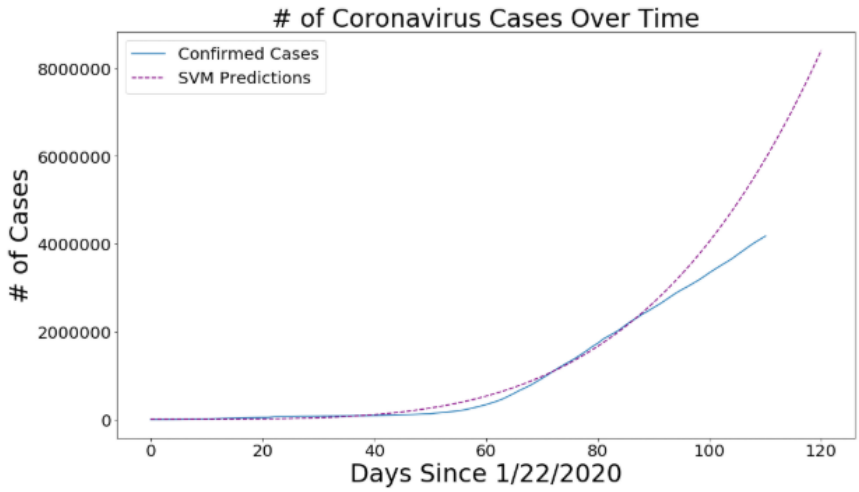


Figure 13

	Date	SVM Predicted # of Confirmed Cases Worldwide
0	05/12/2020	6162348.0
1	05/13/2020	6387212.0
2	05/14/2020	6618180.0
3	05/15/2020	6855363.0
4	05/16/2020	7098870.0
5	05/17/2020	7348813.0
6	05/18/2020	7605304.0
7	05/19/2020	7868456.0
8	05/20/2020	8138385.0
9	05/21/2020	8415205.0

Figure 14

- **Polynomial Regression**

polynomial regression is a form of regression analysis in which the relationship between the independent variable x and the dependent variable y is modeled as an nth degree polynomial in x.

Advantages of using Polynomial Regression:

1. Polynomial provides the best approximation of the relationship between the dependent and independent variables. 2. A broad range of functions can be fit under it. 3. Polynomial fits a wide range of curvature.

The algorithm is divided into the following steps:

1. transform our data for polynomial regression.
2. Use the SVM model to fit the data
3. Plot for Cross-Validation

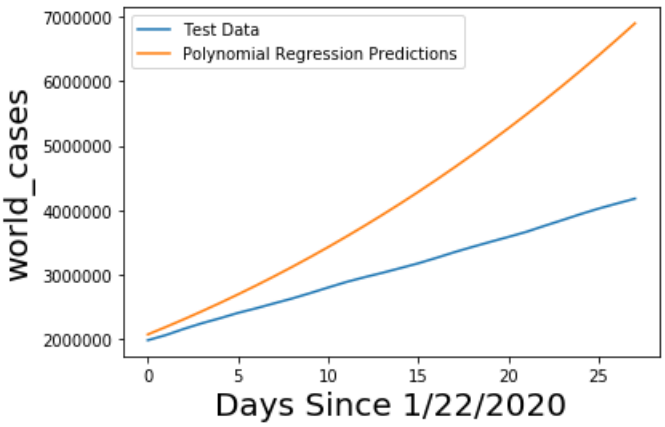


Figure 15

4. Output forecast data

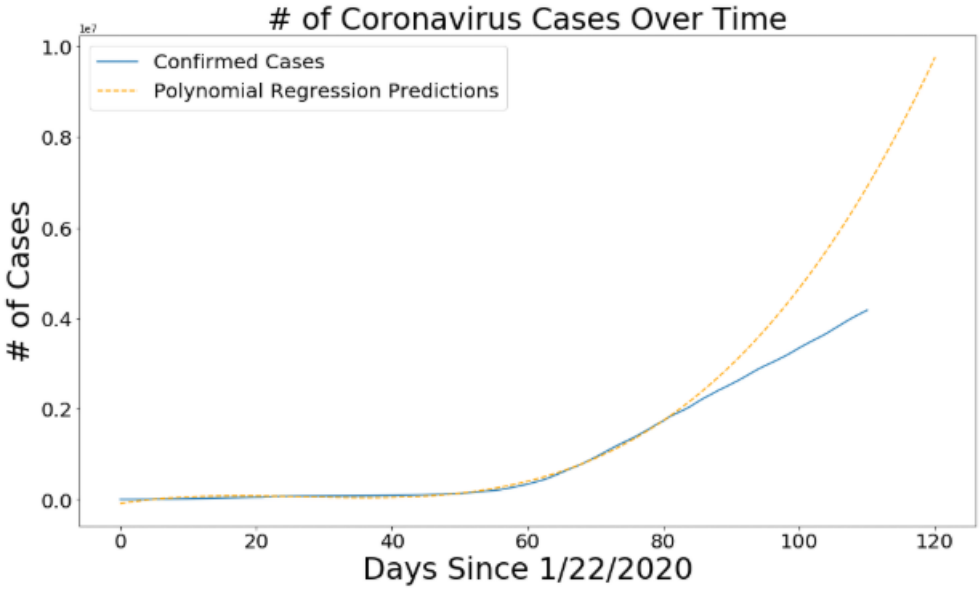


Figure 16

	Date	Polynomial Predicted # of Confirmed Cases Worldwide
0	05/12/2020	7160223.0
1	05/13/2020	7423144.0
2	05/14/2020	7692393.0
3	05/15/2020	7968042.0
4	05/16/2020	8250167.0
5	05/17/2020	8538842.0
6	05/18/2020	8834141.0
7	05/19/2020	9136137.0
8	05/20/2020	9444907.0
9	05/21/2020	9760523.0

Figure 17

- **Linear Regression**

linear regression is a linear approach to modeling the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables). The case of one explanatory variable is called simple linear regression.

Advantages of Linear Regression: 1. Linear Regression performs well when the dataset is linearly separable. We can use it to find the nature of the relationship among the variables. 2. Linear Regression is easier to implement, interpret and very efficient to train. 3. Linear Regression is prone to overfitting but it can be easily avoided using some dimensionality reduction techniques, regularization (L1 and L2) techniques and cross-validation.

The algorithm is divided into the following steps:

1. Plot pairwise relationships in a dataset. You can see that the diagonals are histograms (distribution diagrams) of the individual attributes, while off the diagonals are correlation diagrams between two different attributes.

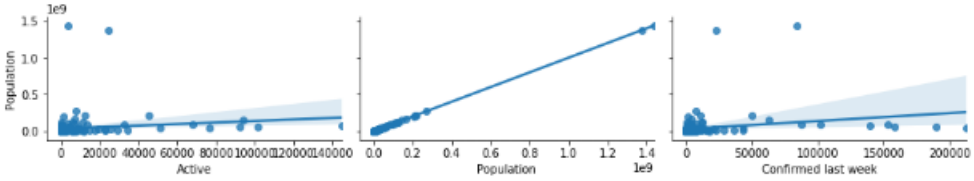


Figure 18

2. Collate feature column
3. Check Data
4. Output the coefficient matrix of the characteristic variable

```
[('Active', 0.044293059250275164), ('Population', 1.0005398121306546e-06), ('Confirmed
last week', -0.009388313933364518)]
[ 963.48728025  150.69168108  155.57588284  112.4603503    55.89722305
 1504.1437114    61.80036031  110.34931814  281.55328447   67.85069461
   56.47301585  101.59587142   86.96120024   84.36358221  105.37407862
  146.07465295   77.68260308  705.82023982   54.87221102   85.54380134
   65.17532053   55.4461459    52.43224606 2306.48435068   84.14462065
   54.86043328  116.27405744   58.65943057   54.52937848  158.85792107
  123.62534615   66.41235444   73.65702183 3765.35909394   61.65705278
   69.65421002  391.43557379   54.72860788   67.31941302   60.42206204
  331.86284948 1264.93743161   54.27362357   54.05165581 1020.64263584
  318.56479576  144.39008879]
```

Figure 19

5. Use the Linear Regression model to fit the data

```
linreg = LinearRegression()
model = linreg.fit(X_train, y_train)
```

Figure 20

6. Plot for Cross-Validation

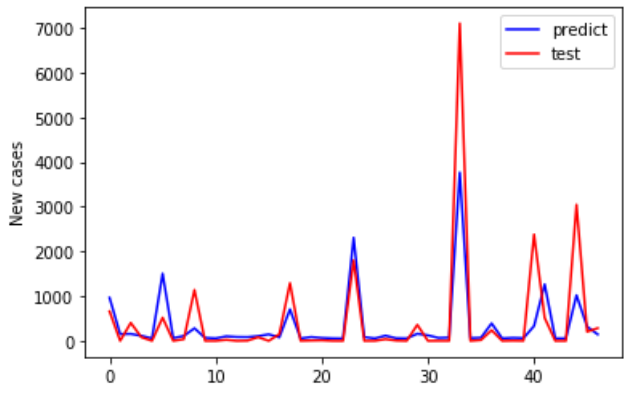


Figure 21

- **Linear model versus the linear model**

	tool	MAE:	MSE:	R Square:
0	SupportVectorMachines(SVM)	6.064162e+05	6.643128e+11	0.538413
1	PolynomialRegression	1.108731e+06	1.871085e+12	0.108595
2	LinearRegression	3.086796e+02	4.829494e+05	-0.071092

Figure 22

In the end, we unified the scores of several linear models together for comparison. By comparing mean_absolute_error,mean_squared_error and R_Square. It can be found that, in general, the model prediction results obtained by polynomial regression are the closest to the real value with the smallest error.

LSTM

LSTM is one of the "Gated RNN" which allows the RNN to coefficient between nodes at greater distances to be changed at different times, and allows the network to "abandon" the information that has been accumulated. The method that is used to train the "LSTM" is called BPTT (backpropagation through time). Back-propagation (BP) calculating the gradient of loss functions which is minimized by update the weight, is one of the algorithms that are used to train neural networks.

The module of Python supports LSTM: scikit-learn and Keras and explanation part of the code is shown below:

1. Separate the data into two rows, one of which contains the values at time t and the other row of which contains the value of t+1. In addition, dataset was normalized by function MinMaxScaler().
2. Set the train size as 30% and split the data into train and test sets.
3. Reshape the input to a structure as [samples, time steps, features]
4. Set the parameters of the LSTM model.
   (1) The number of neurons.
   (2) Iteration times.
   (3) Batch_size: number of samples at each training.
5. Calculate the error and get the training score.
   Train Score: 4435.85 RMSE
   Test Score: 215223.13 RMSE
6. The test result is shown below. Parameters are look_back as 4, batch as 1, epoch as 100, and neuron as 5.

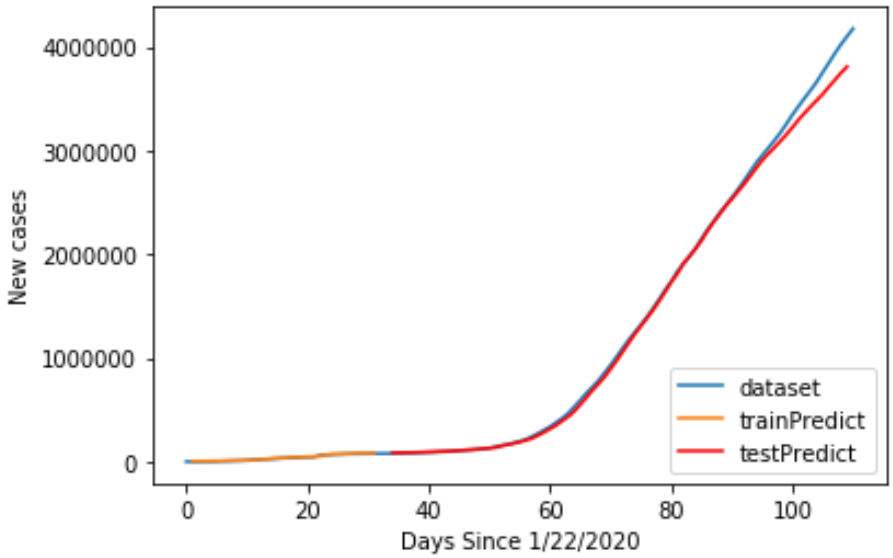


Figure 23

As is shown at the result plotting by the output of the LSTM modeling, it fitted well between the test output and the training output. However, there were still challenges when we worked on codding this algorithm by Python, the most problem of which is that difficulties emerged when we picked up the parameters. Increasing the epoch leads to the fact that there are more regressions, generally making the result more precise but costing more time and PC resources.

Part 2 Bayesian network structure learning analysis

9. BN Structure learning part

1. Knowledge-based graph:

Based on the knowledge and understanding of the spread and control of the Pandemic, we constructed a causal graph represented as DAG (Directed Acyclic Graph) to launch the Bayesian network structure learning shown below.

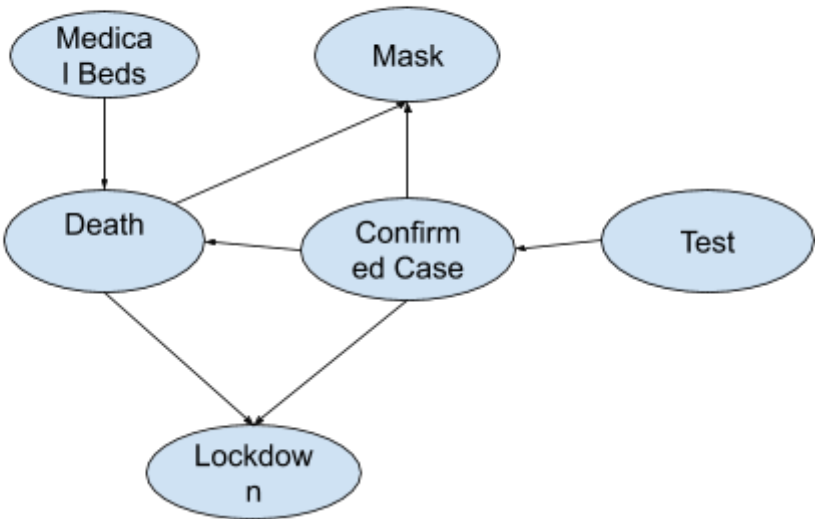


Figure 24 Knowledge-based graph

The nodes are divided into three types as below:

1) Boolean Nodes: Mask, Lockdown, Medical;

Boolean nodes, including Mask and Lockdown, are marked as 1 True and 0 False based on the announced dates of those counties.

2) Numerical Interval nodes: Test, Confirmed, Death;

Nodes "confirmed," "death," and "test" represent the daily increased number of confirmed death and tests performed per million, respectively. They are both divided evenly into ten intervals; we use 1 to 10 to label the intervals.

3) Discrete Numerical Nodes: “Medical Beds” node represented as numeral values;

We used the Coronavirus program by Oxford Martin School in the data processor of each variable. Link: https://ourworldindata.org/coronavirus

2. The answer to the Questions

Question 1: We constructed the causal graph based on the previous understanding of coronavirus. Two members are responsible for the modeling of the causal graph. We used the announcement of WHO and medical papers and the government announcements from PHE. During the drafting, one member holds a different opinion than the medical condition may have a causal relationship with node Confirmed. Therefore, we exchanged mutual knowledge and had a consensus that although increasing confirmed cases do affect the improvement of the medical condition, it had little importance for the causal relation we want to discover in this report, and we delete that association finally.

The SaiyanH graphs in each phase shown as below:

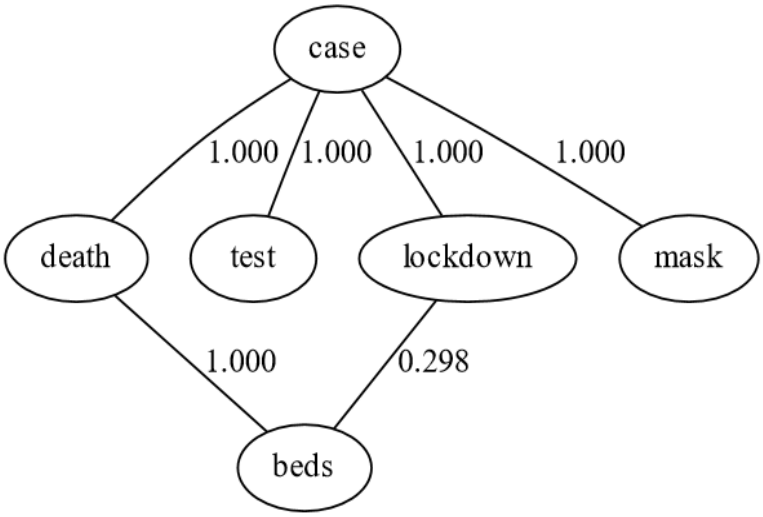


Figure 26

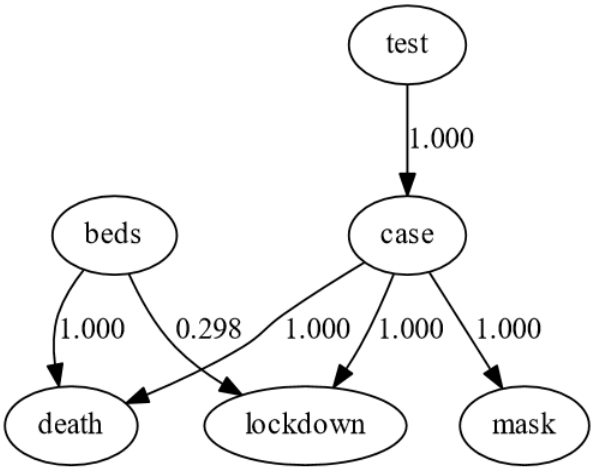


Figure 27

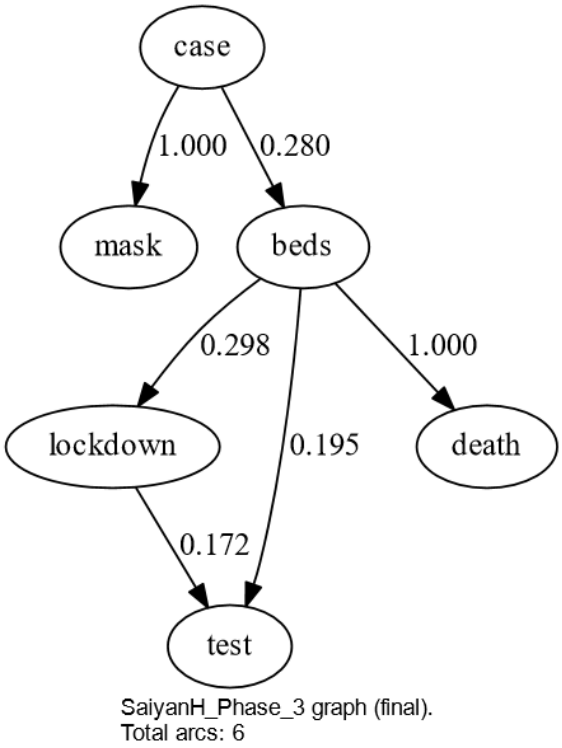


Figure 28

Question 2: The difference between Phase 1 and Phase two is that Phase 1 is association learning, and Phase 2 is about learning constraints. In Phase 1, SaiyanH learned the associations using MMD (Mean/Max/MeanMax Marginal Discrepancy) represent the dependency of each association. In Phase 2, SaiyanH divided the nodes within the EMSG into triples and launched a conditional independence test to classification the triples as three types, Conditional dependence, independence, and insignificance by evaluating the difference between MMD(AB)|C and MMD(AB). Do calculus is used to reassess the un-direct arcs. Finally, directed EMSG is produced.

Question 3: The final Phase is scored-based learning; it is a pruning of the directed EMSG from Phase 2. SaiyanH searches neighboring graphs using BIC as the scoring method. It starts from Hill-Climbing in conjunction with BIC to evaluate the reversed direct of each

orientation and prune the search space if the BIC of reversed arcs greater than the original. It then moves forward to maximize BIC by Tabu search. SaiyanH finally returned a graph that maximizes BIC. If both graphs are the same, it means each dependency is significantly directly, and no other dependencies prevail, or the dependency rate of others is far too not obvious (MMD less than 0.05).

Question 4:

Index	CSV file	Number of rows
1	conditionalDep	4
2	conditonalIndep	4
3	conditional significance	52
4	marginalDep	15

File1 and file2 represent conditional dependencies/ in-dependencies (triples) the algorithm learned in Phase 2. The result means in this case, there are only 4 triple combinations that are conditionally dependent according to the classification rules applied in Phase 2, and only 4 conditionally independent as well.

File3 represents the conditional significance among triple nodes and it indicates rates, 52 is the number of possible combinations minus the constraints we added in the input file.

The rows in file4 indicate the marginal dependencies between every two nodes, hence the number of rows in this file is much bigger than file 1 and 2. For six nodes here, the possible combinations is（6-1）*6/2=15.

Step 3:

```
_____ Evaluation _____
Nodes: 6
Sample size: 500
TrueDAG arcs: 7
TrueDAG independencies: 8
LearnedDAG arcs: 7
LearnedDAG independencies: 8
_____ Confusion matrix stats _____
Arcs discovered (TP): 7.0
Partial arcs discovered (TP*0.5): 0.0
False dependencies discovered (FP): 0.0
Independencies discovered (TN): 8.0
Dependencies not discovered (FN): 0.0. [NOTE: # of edges missed is 0.0]
_____ Stats from metrics and scoring functions _____
Precision score: 1.000
Recall score: 1.000
F1 score: 1.000
SHD score: 0.000
DDM score: 1.000
BSF score: 1.000
# of independent graphical fragments: 1
_____ Inference-based evaluation _____
BIC/MDL score -6012.443
# of free parameters 471
BUILD SUCCESSFUL (total time: 18 seconds)
```

Figure 29

step 4:

```
_____ Training data info _____
Variables: 6
Sample size: 500
_____ Knowledge-based constraints _____
Temporal constraints specified: 6
Directed constraints specified: 5
_____ Structure learning _____
Running SaiyanH with settings:
    a) Associational score: MeanMax [Absolute]
    b) Conditional independence pruning: true
    c) Faithfulness condition pruning: true
    d) TABU search max escape attempts: V(V-1)
Entering Phase 1 [EMST graph]...
marginalDep.csv saved.
Phase 1 completed.
Entering Phase 2 [constraint-based learning]...
conditionalDep.csv saved.
conditionalIndep.csv saved.
conditionalInsignificance.csv saved.
Phase 2 completed.
Entering Phase 3 [score-based learning]...
case->mask[label="1.000"];case->beds[label="0.280"];lockdown->test[label="0.172"];beds-
Phase 3 completed.
Arcs randomised during phase 2 constraint-based learning: 0
Structure learning elapsed time: 2 seconds total (Phase 1 = 1 secs, Phase 2 = 2 secs).
```

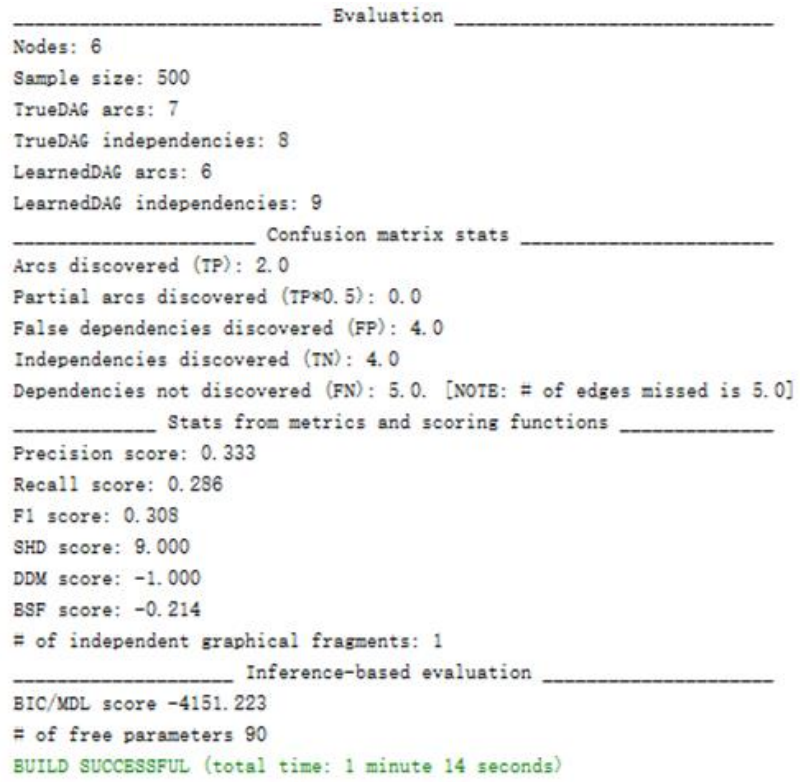


Figure 30

Question 5: F1 is lower than the reference value. The precision score and recall score that influenced the result is too low. The lower score indicates a more inaccurate chart. SHD is higher than the reference value. The higher score indicates a perfect match between learned and true graphs. BSF is lower than the reference value because the BSF score is on the basis that all direct independencies have been discovered. In this paper, there are fewer nodes and edges. It cannot balance the relationship between edge and point better. So, the value is lower. The results are in line with expectations

Question 6: The causal model in this thesis has six nodes, seven true edges, 90 free parameters, and 500 samples. The run time is determined by the number of nodes and the size of the data. This is not a very complex model, so it takes less runtime than in Table 2.

Question 7: The BIC/MDL score generated in Step 4 is lower than the data in Step 3. The first item of the BIC score is the log-likelihood of the model. Measure how well the structure fits the data. The second is the penalty term of model complexity. Prevent the transition fit between data and structure. The BIC will be reduced by learning the structure first and then evaluating it. Make it choose a lower balance point between complexity and accuracy of network structure.

Question 8: The # of free parameters generated in Step 4 is lower than the data in Step 3. Variables, parents, and states that affect free parameters. Because the fourth step is to learn the structure first and then evaluate it, the number of arcs in the DAGlearned.csv is different. So, the result is in agreement with your initial expectations.

10.Successes and Challenges

In this paper, novel coronavirus transmission is evaluated by linear models, neural networks, and Bayesian models. All models are based on actual data from the global novel coronavirus, which has stimulated varying degrees of interdiction around the world. Based on these data, the blockade should continue to try to 'level the curve'.

Although the improved model can make an ideal trend analysis of the epidemic situation, it still deviates from the real situation in the later stage of the epidemic. Three reasons mainly cause it. First, the model is too simplified, and the real situation is over idealized. The infection of this epidemic is not only transmitted by the infected people, but also can be used as a source of infection for some susceptible people as long as they carry pathogens, but this part of the population is not reflected in the establishment of the model. Second, the epidemic in Europe is still on the rise, and the real data is insufficient. Also, the data caused by the government's neglect in the early stage is abnormal. We have no way to make a good fit before the control. Similarly, we do not know after control. Third, the population base of all countries is enormous. This is an excellent challenge for the establishment of the model because the general model requires a fixed population and no population communication.

11.Business Applications

Since the COVID-19 outbreak, research institutions and academics around the world have published a variety of model predictions, some of which seem alarmist and compelling. Some people have questioned some of the predictions made by some agencies, which may lead to some public fears about the future. This requires researchers to be clear about the conditions under which their predictions were made when publishing their results. Any prediction based on a mathematical model should not be independent of conditions. It is based on these conditions that the public can understand the real meaning behind some seemingly "terrible" predictions. Hence, a detailed and clear explanation is the best way to avoid causing public panic.

The results of this project can serve as a warning that new easing of the blockade could have catastrophic consequences, leading to more cases or the second wave of outbreaks.

Mathematical models were used to predict the epidemic situation, which will also be used to provide scientific prevention and control of the epidemic, including epidemic monitoring and analysis, virus tracking, patient tracking, personnel flow, and community management. During the COVID-19 epidemic prevention period, big data not only helped the government make scientific decisions and optimize the allocation of resources in the field of public health management but also enabled the public to keep abreast of the development of the epidemic and take an active and scientific approach to epidemic prevention.

12.Concluding remarks

The key findings of this analysis based on the global novel coronavirus data for 2020 are summaries as follows:

- The Linear model predicts the peak time of the novel coronavirus.
- Through BN structure learning, we analyzed the causal relationship among the mitigation factors, the confirmed case, and the death rate.
- Due to inconsistent data updates and different data quality in different regions, the predicted results will change with time.
- LSTM algorithm was used to predict for the peak time of the novel coronavirus.
- Different models have different data processing capabilities, in the process of selecting the model, the idea of checking and verifying should be applied to verify the simulation

effect of the model.

- A detailed and clear explanation is the best way to avoid causing public panic.
- By visualizing the epidemic data, we can find that the epidemic has been contained to some extent, but from country to country. Countries should strengthen cooperation and share more resources on the existing basis. Only with the concerted efforts of all countries can we finally overcome the epidemic.

13.References

1) Chinazzi M, Davis J T, Ajelli M, et al. (2020) The effect of travel restrictions on the spread of the 2019 novel coronavirus (COVID-19) outbreak[J]. Science, 368(6489): 395-400.
2) Flaxman, Seth & Mishra, et al. (2020). Estimating the number of infections and the impact of non-pharmaceutical interventions on COVID-19 in European countries: technical description update, Imperial College London,
3) Wells C R, Sah P, Moghadas S M, et al. 2020, the impact of international travel and borer control measures on the global spread of the novel 2019 coronavirus outbreak[J]. Proceedings of the National Academy of Sciences, 117(13): 7504-7509.
4) McKinney W. pandas, 2011, a foundational Python library for data analysis and statistics[J]. Python for High Performance and Scientific Computing, 14(9).
5) Géron A. 2019, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems[M]. O'Reilly Media.
6) Chen S H, Pollino C A., 2012, Good practice in Bayesian network modeling [J]. Environmental Modelling & Software, 37: 134-145.
7) Garreta, R. & Moncecchi, G. 2013, Learning scikit-learn: machine learning in Python, 1st edition, Packt Publishing, Birmingham
8) Li Shuzhi. 2010, Research on Bayesian Network Structure Scoring Function, Chinese Conference on Pattern Recognition(CCPR 2010), Chongqing
9) Rossant Cyrille, 2013, Learning IPython for Interactive Computing and Data Visualization, Packt Publishing Ltd, UK.
10) Švaňa M, Němec R, 2018, Comparison of Linear SVM Algorithm Implementations in Python for Solving an Author Identification Problem, Proceedings of the 21st International Conference on Information Technology for Practice (IT for practice 2018), C88, Czech

14.Appendices

Code and Data: The full code and partially downloaded **data** is available on ***GitHub*** through the following link: https://github.com/apzzzzzzzz/DA_project

- **Importing the libraries, we need**

In [1]:
```
#importing the libraries that we use in our project

import numpy as np
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors
import pandas as pd
import random
import math
#import time
#import xgboost
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
from sklearn.preprocessing import MinMaxScaler
from sklearn.svm import SVR
#from sklearn.svm import SVC
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn.metrics import r2_score
#from sklearn.metrics import classification_report
#from sklearn.metrics import confusion_matrix
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
#import plotly.express as px
import datetime
import operator
import folium
```

```
Using TensorFlow backend.
```

- **Importing datasets and creating the DataFrames**

```
#Creating the DataFrames
confirmed_df = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/
deaths_df = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/mas
recoveries_df = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19
population_data=pd.read_csv('C:/Users/1/Desktop/数据集/DATA/population_by_country_2020
full_table = pd.read_csv('C:/Users/1/Desktop/数据集/DATA/full_table_clean.csv',parse_d
country_wise_pop_weekly_latest = pd.read_csv('C:/Users/1/Desktop/数据集/DATA/country_w
latest_data = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/m
```

- **The structure of each DataFrame**

```
#The structure of country_wise_pop_weekly_latest.csv
country_wise_pop_weekly_latest.head(10)
```

	Unnamed: 0	Country/Region	Confirmed	Deaths	Recovered	Active	New cases	Deaths / 100 Cases	Recovered / 100 Cases	Deaths / 100 Recovered	Population	Yearly Change	Net Change	Density (P/Km²)	La A (K
0	0	Afghanistan	2171	64	260	1847	232	2.95	11.98	24.62	38742911.0	2.33 %	886592.0	60.0	65286
1	1	Albania	773	31	470	272	7	4.01	60.80	6.60	2878420.0	-0.11 %	-3120.0	105.0	2740
2	2	Algeria	4006	450	1779	1777	158	11.23	44.41	25.30	43685618.0	1.85 %	797990.0	18.0	238174
3	3	Andorra	745	42	468	235	2	5.64	62.82	8.97	77240.0	0.16 %	123.0	164.0	47
4	4	Angola	27	2	7	18	0	7.41	25.93	28.57	32644783.0	3.27 %	1040977.0	26.0	124670
5	5	Antigua and Barbuda	24	3	11	10	0	12.50	45.83	27.27	97764.0	0.84 %	811.0	223.0	44
6	6	Argentina	4428	218	1256	2954	143	4.92	28.36	17.36	45111229.0	0.93 %	415097.0	17.0	273669
7	7	Armenia	2066	32	929	1105	134	1.55	44.97	3.44	2962137.0	0.19 %	5512.0	104.0	2847
8	8	Australia	6766	93	5742	931	14	1.37	84.87	1.62	25439164.0	1.18 %	296686.0	3.0	768230
9	9	Austria	15452	584	12907	1961	50	3.78	83.53	4.52	8996022.0	0.57 %	51296.0	109.0	8240

In [4]:
```
#The structure of time_series_covid19_confirmed_global.csv
confirmed_df.head(2)
```

Out[4]:

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	...	5/2/20	5/3/20	5/4/20	5/5/20	5/6/20	5/7/20	5/8/20
0	NaN	Afghanistan	33.0000	65.0000	0	0	0	0	0	0	...	2469	2704	2894	3224	3392	3563	3778
1	NaN	Albania	41.1533	20.1683	0	0	0	0	0	0	...	789	795	803	820	832	842	850

2 rows × 115 columns

In [5]:
```
#The structure of population_by_country_2020.csv
population_data.head(2)
```

Out[5]:

	Country (or dependency)	Population (2020)	Yearly Change	Net Change	Density (P/Km²)	Land Area (Km²)	Migrants (net)	Fert. Rate	Med. Age	Urban Pop %	World Share
0	China	1438207241	0.39 %	5540090	153	9388211	-348399.0	1.7	38	61 %	18.47 %
1	India	1377233523	0.99 %	13586631	464	2973190	-532687.0	2.2	28	35 %	17.70 %

- **Check values**

```
#Data processing
confirmed_df.describe(include = 'object')
```

	Province/State	Country/Region
count	82	266
unique	82	187
top	Hong Kong	China
freq	1	33

```
a = full_table.Date.value_counts().sort_index()
print('days covered:',len(a))
print('The first date is:',a.index[0])
print('The last date is:',a.index[-1])
```

```
days covered: 100
The first date is: 2020-01-22 00:00:00
The last date is: 2020-04-30 00:00:00
```

```
full_table.isnull().sum()
confirmed_df.isnull().sum()
deaths_df.isnull().sum()
recoveries_df.isnull().sum()
population_data.isnull().sum()
latest_data.isnull().sum()
```

```
FIPS              275
Admin2            272
Province_State    183
Country_Region      0
```

- **Removing Duplicates**

```
#remove duplicate
full_table.duplicated().sum()
confirmed_df.duplicated().sum()
deaths_df.duplicated().sum()
recoveries_df.duplicated().sum()
population_data.duplicated().sum()
latest_data.duplicated().sum()
```

- **Adding columns**

```
full_table['Active'] = full_table['Confirmed'] - full_table['Deaths'] - full_table['Recovered']
day_wise['Deaths / 100 Cases'] = round((day_wise['Deaths']/day_wise['Confirmed'])*100, 2)
day_wise['Recovered / 100 Cases'] = round((day_wise['Recovered']/day_wise['Confirmed'])*100, 2)
day_wise['Deaths / 100 Recovered'] = round((day_wise['Deaths']/day_wise['Recovered'])*100, 2)
country_wise['Deaths / 100 Cases'] = round((country_wise['Deaths']/country_wise['Confirmed'])*100, 2)
country_wise['Recovered / 100 Cases'] = round((country_wise['Recovered']/country_wise['Confirmed'])*100, 2)
country_wise['Deaths / 100 Recovered'] = round((country_wise['Deaths']/country_wise['Recovered'])*100, 2)
country_wise['Cases / Million People'] = round((country_wise['Confirmed'] / country_wise['Population']) * 1000000)
```

- **Dealing with Missing Values**

```
full_table[['Province/State']] = full_table[['Province/State']].fillna('')
full_table[['Confirmed', 'Deaths', 'Recovered', 'Active']] = full_table[['Confirmed', 'Deaths', 'Recovered', 'Active']].fillna(0)
```

```
full_grouped = full_grouped.fillna(0)
```

- **Checking data type**

```
full_table['Recovered'] = full_table['Recovered'].astype(int)
```

- **data combination**

```
full_grouped = full_table.groupby(['Date', 'Country/Region'])['Confirmed', 'Deaths', 'Recovered', 'Active
day_wise = full_grouped.groupby('Date')['Confirmed', 'Deaths', 'Recovered', 'Ac
```

- **merged data**

```
# merged data
country_wise = pd.merge(country_wise,population_data, on='Country/Region', how='left')
```

- **Save the new data table.**

```
country_wise.to_csv('C:/Users/1/Desktop/数据集/country_wise_pop_weekly_latest.csv')
```

- **simplify data by choosing columns**

```
#simplify data by choosing colums

confirmed = confirmed_df.loc[:, cols[4]:cols[-1]]
deaths = deaths_df.loc[:, cols[4]:cols[-1]]
recoveries = recoveries_df.loc[:, cols[4]:cols[-1]]
```

- **Create dataset**

```
# calculate rates
mortality_rate.append(death_sum/confirmed_sum)
recovery_rate.append(recovered_sum/confirmed_sum)

# case studies
china_cases.append(confirmed_df[confirmed_df['Country/Region']=='China'][i].sum())
italy_cases.append(confirmed_df[confirmed_df['Country/Region']=='Italy'][i].sum())
us_cases.append(confirmed_df[confirmed_df['Country/Region']=='US'][i].sum())
spain_cases.append(confirmed_df[confirmed_df['Country/Region']=='Spain'][i].sum())
france_cases.append(confirmed_df[confirmed_df['Country/Region']=='France'][i].sum())
germany_cases.append(confirmed_df[confirmed_df['Country/Region']=='Germany'][i].sum())
uk_cases.append(confirmed_df[confirmed_df['Country/Region']=='United Kingdom'][i].sum())
russia_cases.append(confirmed_df[confirmed_df['Country/Region']=='Russia'][i].sum())

china_deaths.append(deaths_df[deaths_df['Country/Region']=='China'][i].sum())
italy_deaths.append(deaths_df[deaths_df['Country/Region']=='Italy'][i].sum())
us_deaths.append(deaths_df[deaths_df['Country/Region']=='US'][i].sum())
spain_deaths.append(deaths_df[deaths_df['Country/Region']=='Spain'][i].sum())
france_deaths.append(deaths_df[deaths_df['Country/Region']=='France'][i].sum())
germany_deaths.append(deaths_df[deaths_df['Country/Region']=='Germany'][i].sum())
uk_deaths.append(deaths_df[deaths_df['Country/Region']=='United Kingdom'][i].sum())
russia_deaths.append(deaths_df[deaths_df['Country/Region']=='Russia'][i].sum())

china_recoveries.append(recoveries_df[recoveries_df['Country/Region']=='China'][i].sum())
italy_recoveries.append(recoveries_df[recoveries_df['Country/Region']=='Italy'][i].sum())
us_recoveries.append(recoveries_df[recoveries_df['Country/Region']=='US'][i].sum())
spain_recoveries.append(recoveries_df[recoveries_df['Country/Region']=='Spain'][i].sum())
france_recoveries.append(recoveries_df[recoveries_df['Country/Region']=='France'][i].sum())
germany_recoveries.append(recoveries_df[recoveries_df['Country/Region']=='Germany'][i].sum())
uk_recoveries.append(recoveries_df[recoveries_df['Country/Region']=='United Kingdom'][i].sum())
russia_recoveries.append(recoveries_df[recoveries_df['Country/Region']=='Russia'][i].sum())
```

- **Create daily_increase function and fill data**

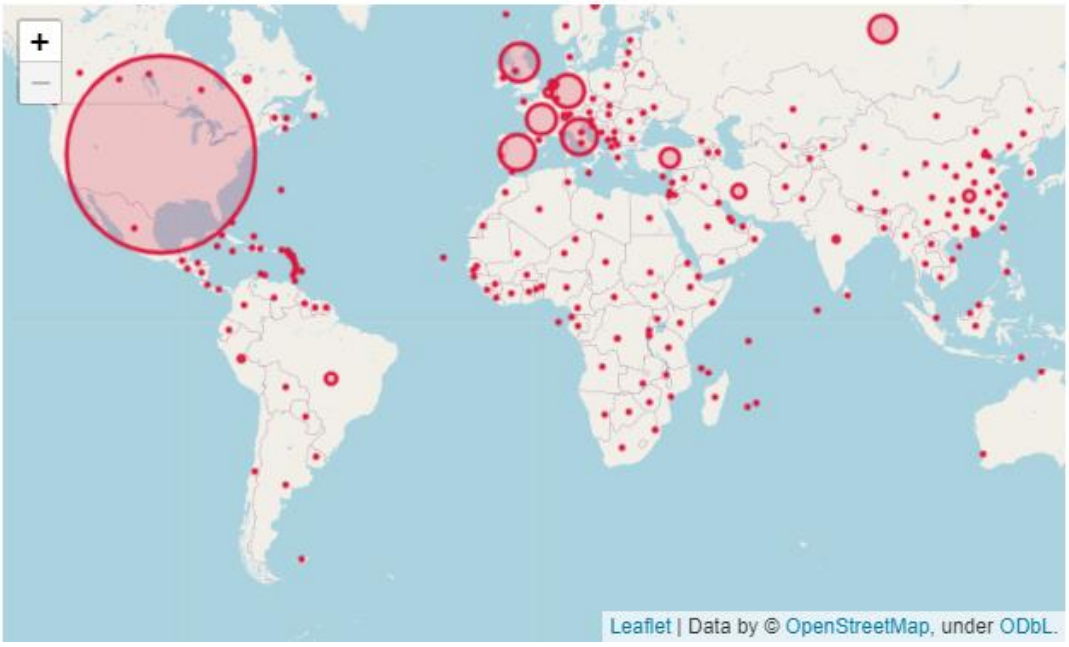
```
def daily_increase(data):
    d = []
    for i in range(len(data)):
        if i == 0:
            d.append(data[0])
        else:
            d.append(data[i]-data[i-1])
    return d

# confirmed cases
world_daily_increase = daily_increase(world_cases)
china_daily_increase = daily_increase(china_cases)
italy_daily_increase = daily_increase(italy_cases)
us_daily_increase = daily_increase(us_cases)
spain_daily_increase = daily_increase(spain_cases)
france_daily_increase = daily_increase(france_cases)
germany_daily_increase = daily_increase(germany_cases)
uk_daily_increase = daily_increase(uk_cases)
russia_daily_increase = daily_increase(russia_cases)

# deaths
world_daily_death = daily_increase(total_deaths)
china_daily_death = daily_increase(china_deaths)
italy_daily_death = daily_increase(italy_deaths)
us_daily_death = daily_increase(us_deaths)
spain_daily_death = daily_increase(spain_deaths)
france_daily_death = daily_increase(france_deaths)
germany_daily_death = daily_increase(germany_deaths)
uk_daily_death = daily_increase(uk_deaths)
russia_daily_death = daily_increase(russia_deaths)

# recoveries
world_daily_recovery = daily_increase(total_recovered)
china_daily_recovery = daily_increase(china_recoveries)
italy_daily_recovery = daily_increase(italy_recoveries)
us_daily_recovery = daily_increase(us_recoveries)
spain_daily_recovery = daily_increase(spain_recoveries)
france_daily_recovery = daily_increase(france_recoveries)
germany_daily_recovery = daily_increase(germany_recoveries)
uk_daily_recovery = daily_increase(uk_recoveries)
russia_daily_recovery = daily_increase(russia_recoveries)
```

- **Create country_plot and plot_predictions function**

```
def country_plot(x, y1, y2, y3, y4, country):
    plt.figure(figsize=(16, 9))
    plt.plot(x, y1)
    plt.title('{} Confirmed Cases'.format(country), size=30)
    plt.xlabel('Days Since 1/22/2020', size=30)
    plt.ylabel('# of Cases', size=30)
    plt.xticks(size=20)
    plt.yticks(size=20)
    plt.show()

    plt.figure(figsize=(16, 9))
    plt.bar(x, y2)
    plt.title('{} Daily Increases in Confirmed Cases'.format(country), size=30)
    plt.xlabel('Days Since 1/22/2020', size=30)
    plt.ylabel('# of Cases', size=30)
    plt.xticks(size=20)
    plt.yticks(size=20)
    plt.show()

    plt.figure(figsize=(16, 9))
    plt.bar(x, y3)
    plt.title('{} Daily Increases in Deaths'.format(country), size=30)
    plt.xlabel('Days Since 1/22/2020', size=30)
    plt.ylabel('# of Cases', size=30)
    plt.xticks(size=20)
    plt.yticks(size=20)
    plt.show()

    plt.figure(figsize=(16, 9))
    plt.bar(x, y4)
    plt.title('{} Daily Increases in Recoveries'.format(country), size=30)
    plt.xlabel('Days Since 1/22/2020', size=30)
    plt.ylabel('# of Cases', size=30)
    plt.xticks(size=20)
    plt.yticks(size=20)
    plt.show()

def plot_predictions(x, y, pred, algo_name, color):
    plt.figure(figsize=(16, 9))
    plt.plot(x, y)
    plt.plot(future_forcast, pred, linestyle='dashed', color=color)
    plt.title('# of Coronavirus Cases Over Time', size=30)
    plt.xlabel('Days Since 1/22/2020', size=30)
    plt.ylabel('# of Cases', size=30)
    plt.legend(['Confirmed Cases', algo_name], prop={'size': 20})
    plt.xticks(size=20)
    plt.yticks(size=20)
    plt.show()
```

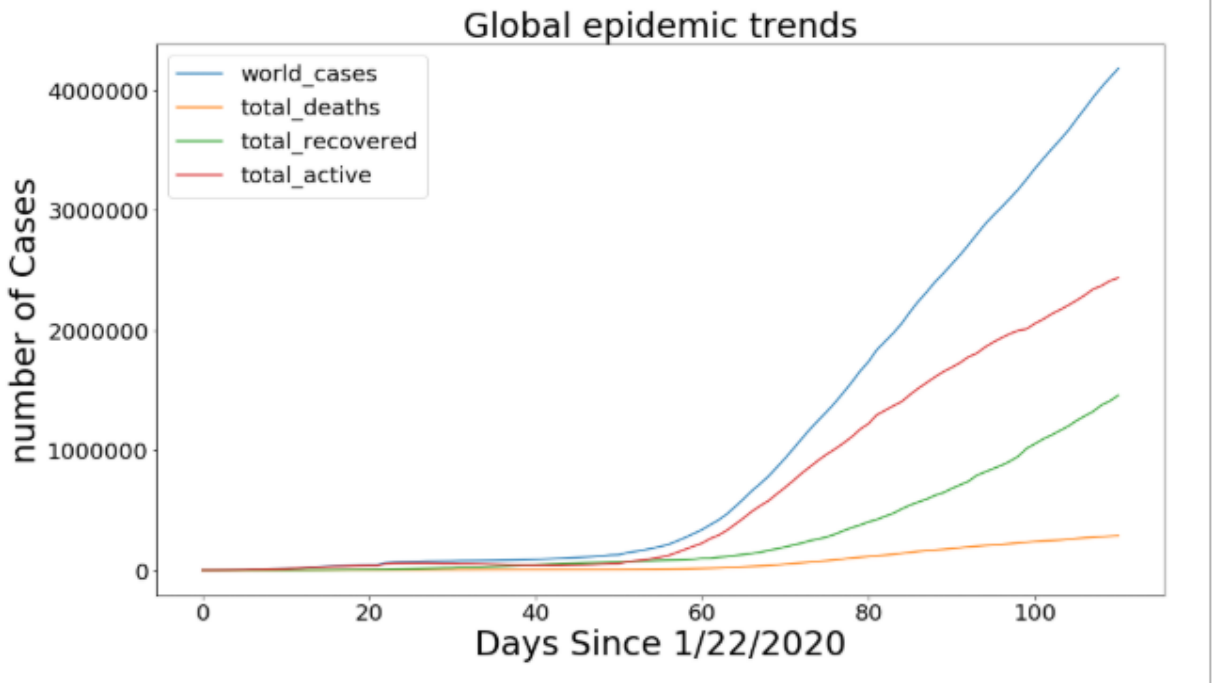
- **World map**

```
temp = full_table[full_table['Date'] == max(full_table['Date'])]

heatmap = folium.Map(min_zoom=2, max_zoom=4, zoom_start=2)

for i in range(0, len(temp)):
    folium.Circle(
        location=[temp.iloc[i]['Lat'], temp.iloc[i]['Long']],
        color='crimson', fill='crimson',
        radius=temp.iloc[i]['Confirmed']**1.05).add_to(heatmap)
heatmap
```



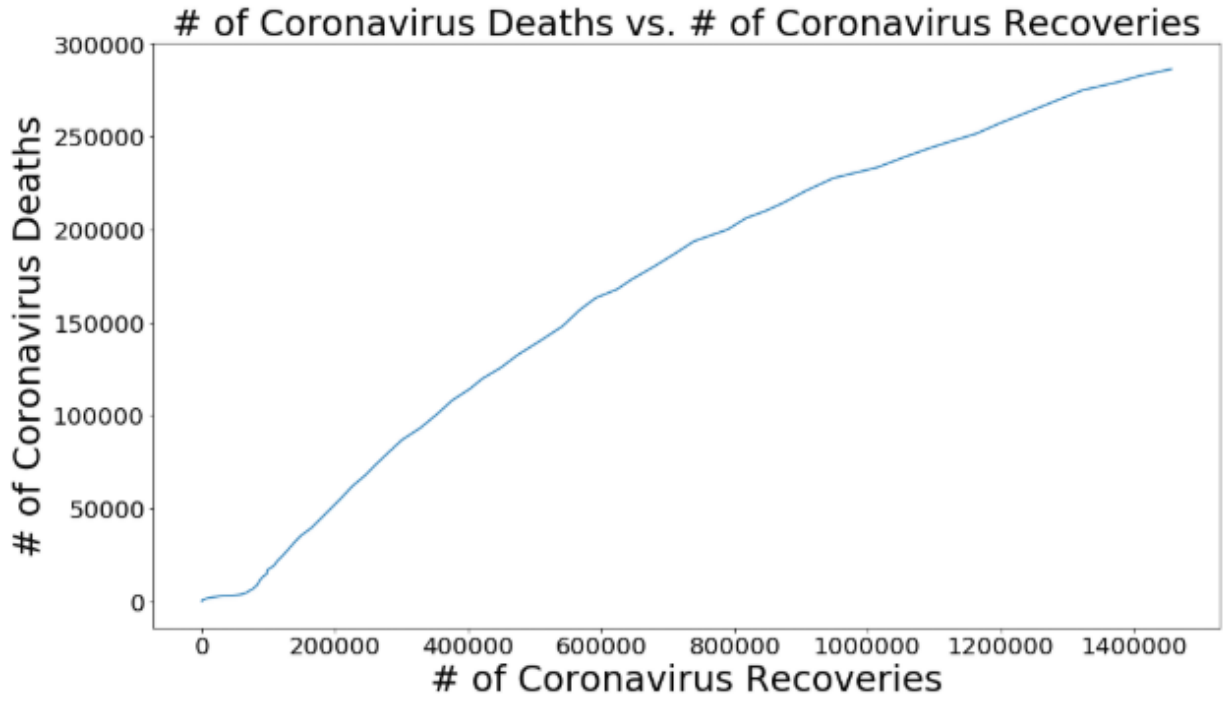
- **Global epidemic trends**

```
plt.figure(figsize=(16, 9))
plt.plot(adjusted_dates, world_cases)
plt.plot(adjusted_dates, total_deaths)
plt.plot(adjusted_dates, total_recovered)
plt.plot(adjusted_dates, total_active)
plt.legend(['world_cases', 'total_deaths', 'total_recovered', 'total_active'], prop={'size': 20})
plt.xlabel('Days Since 1/22/2020', size=30)
plt.ylabel('number of Cases', size=30)
plt.xticks(size=20)
plt.yticks(size=20)
plt.title('Global epidemic trends', size=30)
plt.show()
```



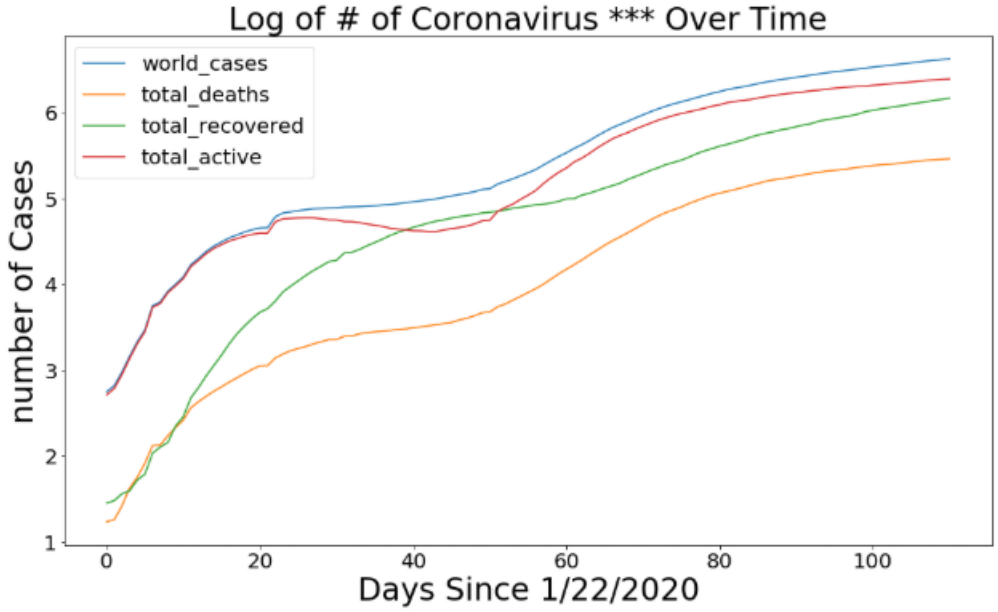
- **Deaths vs. Recoveries**

```
plt.figure(figsize=(16, 9))
plt.plot(total_recovered, total_deaths)
plt.title('# of Coronavirus Deaths vs. # of Coronavirus Recoveries', size=30)
plt.xlabel('# of Coronavirus Recoveries', size=30)
plt.ylabel('# of Coronavirus Deaths', size=30)
plt.xticks(size=20)
plt.yticks(size=20)
plt.show()
```



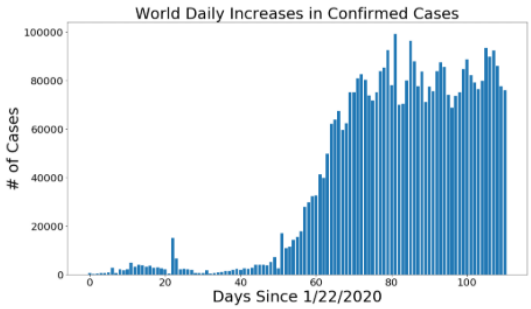
- **Log of Coronavirus Over Time**

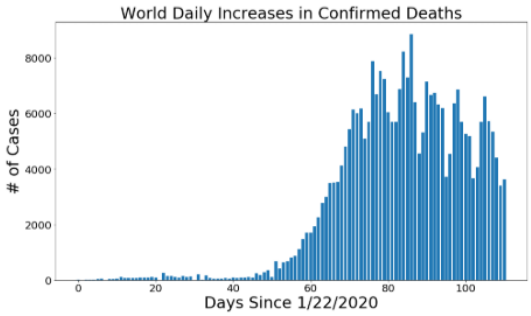
```
plt.figure(figsize=(16, 9))
plt.plot(adjusted_dates, np.log10(world_cases))
plt.plot(adjusted_dates, np.log10(total_deaths))
plt.plot(adjusted_dates, np.log10(total_recovered))
plt.plot(adjusted_dates, np.log10(total_active))
plt.title('Log of # of Coronavirus *** Over Time', size=30)
plt.xlabel('Days Since 1/22/2020', size=30)
plt.ylabel('number of Cases', size=30)
plt.legend(['world_cases', 'total_deaths', 'total_recovered', 'total_active'], prop={'size': 20})
plt.xticks(size=20)
plt.yticks(size=20)
plt.show()
```

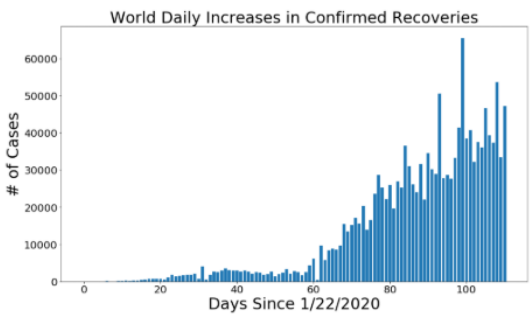


- **World Daily Increases**

```
plt.figure(figsize=(16, 9))
plt.bar(adjusted_dates, world_daily_increase)
plt.title('World Daily Increases in Confirmed Cases',
plt.xlabel('Days Since 1/22/2020', size=30)
plt.ylabel('# of Cases', size=30)
plt.xticks(size=20)
plt.yticks(size=20)
plt.show()

plt.figure(figsize=(16, 9))
plt.bar(adjusted_dates, world_daily_death)
plt.title('World Daily Increases in Confirmed Deaths',
plt.xlabel('Days Since 1/22/2020', size=30)
plt.ylabel('# of Cases', size=30)
plt.xticks(size=20)
plt.yticks(size=20)
plt.show()

plt.figure(figsize=(16, 9))
plt.bar(adjusted_dates, world_daily_recovery)
plt.title('World Daily Increases in Confirmed Recover
plt.xlabel('Days Since 1/22/2020', size=30)
plt.ylabel('# of Cases', size=30)
plt.xticks(size=20)
plt.yticks(size=20)
plt.show()
```

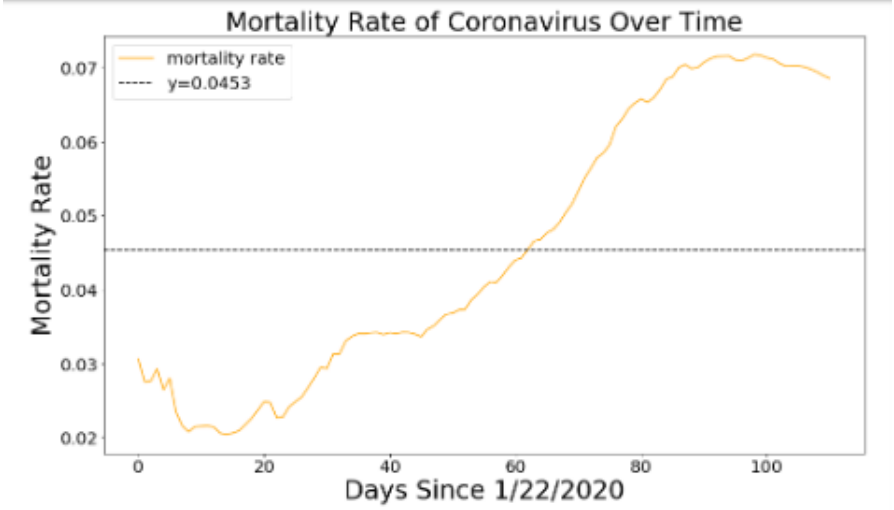






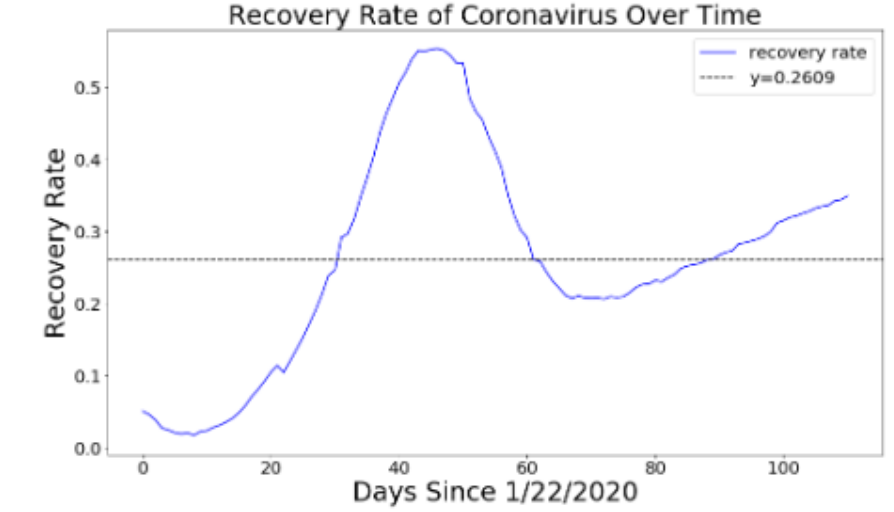
- **Mortality Rate of Coronavirus Over Time**

```
mean_mortality_rate = np.mean(mortality_rate)
plt.figure(figsize=(16, 9))
plt.plot(adjusted_dates, mortality_rate, color='orange')
plt.axhline(y = mean_mortality_rate,linestyle='—', color='black')
plt.title('Mortality Rate of Coronavirus Over Time', size=30)
tep=np.round(mean_mortality_rate,4)
plt.legend(['mortality rate', 'y='+str(tep)], prop={'size': 20})
plt.xlabel('Days Since 1/22/2020', size=30)
plt.ylabel('Mortality Rate', size=30)
plt.xticks(size=20)
plt.yticks(size=20)
plt.show()
```



- **Recovery Rate of Coronavirus Over Time**

```
mean_recovery_rate = np.mean(recovery_rate)
plt.figure(figsize=(16, 9))
plt.plot(adjusted_dates, recovery_rate, color='blue')
plt.axhline(y = mean_recovery_rate,linestyle='—', color='black')
plt.title('Recovery Rate of Coronavirus Over Time', size=30)
tep=np.round(mean_recovery_rate,4)
plt.legend(['recovery rate', 'y='+str(tep)], prop={'size': 20})
plt.xlabel('Days Since 1/22/2020', size=30)
plt.ylabel('Recovery Rate', size=30)
plt.xticks(size=20)
plt.yticks(size=20)
plt.show()
```



- **number of cases per country/region**

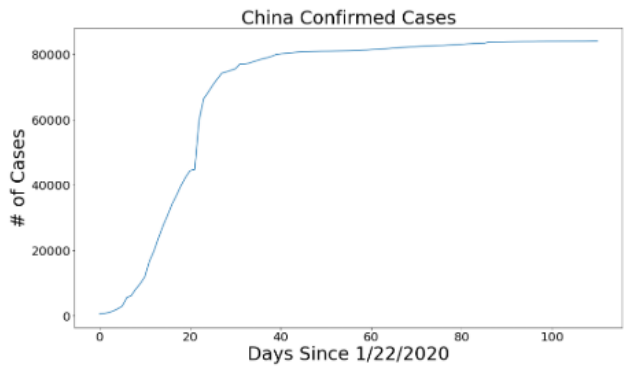
```
unique_countries =  list(latest_data['Country_Region'].unique())
country_confirmed_cases = []
country_death_cases = []
country_active_cases = []
country_recovery_cases = []
country_mortality_rate = []

no_cases = []
for i in unique_countries:
    cases = latest_data[latest_data['Country_Region']==i]['Confirmed'].sum()
    if cases > 0:
        country_confirmed_cases.append(cases)
    else:
        no_cases.append(i)

for i in no_cases:
    unique_countries.remove(i)

# sort countries by the number of confirmed cases
unique_countries = [k for k, v in sorted(zip(unique_countries, country_confir
for i in range(len(unique_countries)):
    country_confirmed_cases[i] = latest_data[latest_data['Country_Region']==ur
    country_death_cases.append(latest_data[latest_data['Country_Region']==uniq
    country_recovery_cases.append(latest_data[latest_data['Country_Region']==u
    country_active_cases.append(country_confirmed_cases[i] - country_death_cas
    country_mortality_rate.append(country_death_cases[i]/country_confirmed_cas
```

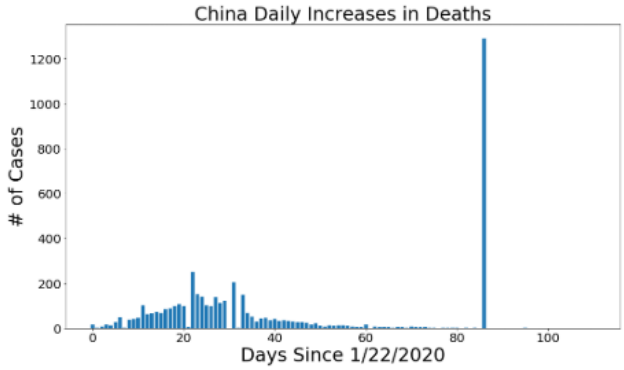
```
country_df = pd.DataFrame({'Country Name': unique_countries, 'Number of Confir
                           'Number of Deaths': country_death_cases, 'Number of
                           'Number of Active Cases' : country_active_cases,
                           'Mortality Rate': country_mortality_rate})
# number of cases per country/region

country_df.style.background_gradient(cmap='Greens')
```

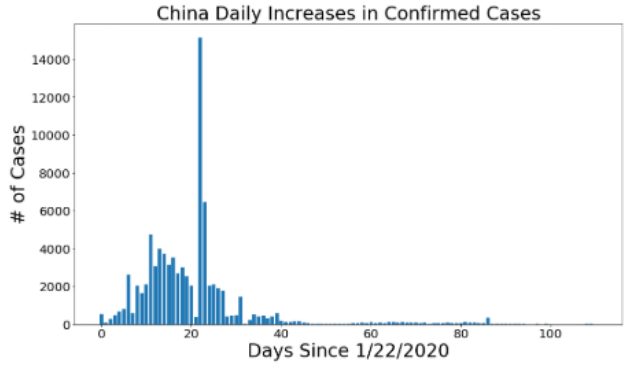
	Country Name	Number of Confirmed Cases	Number of Deaths	Number of Recoveries	Number of Active Cases	Mortality Rate
0	US	1132539	66369	175382	890788	0.058602
1	Spain	216582	25100	117248	74234	0.115891
2	Italy	209328	28710	79914	100704	0.137153
3	United Kingdom	183500	28205	896	154399	0.153706
4	France	168518	24763	50663	93092	0.146946
5	Germany	164967	6812	129000	29155	0.041293
6	Turkey	124375	3336	58259	62780	0.026822
7	Russia	124054	1222	15013	107819	0.009851
8	Brazil	97100	6761	40937	49402	0.069629
9	Iran	96448	6156	77350	12942	0.063827
10	China	83959	4637	78586	736	0.055229
11	Canada	57926	3684	23814	30428	0.063598
12	Belgium	49517	7765	12211	29541	0.156815
13	Peru	42534	1200	12434	28900	0.028213
14	Netherlands	40434	5003	138	35293	0.123733
15	India	39699	1323	10819	27557	0.033326
16	Switzerland	29817	1762	24200	3855	0.059094
17	Ecuador	27464	1371	2132	23961	0.049920
18	Saudi Arabia	25459	176	3765	21518	0.006913
19	Portugal	25190	1023	1671	22496	0.040611
20	Mexico	22088	2061	12377	7650	0.093309
21	Sweden	22082	2669	1005	18408	0.120868
22	Ireland	21176	1286	13386	6504	0.060729
23	Pakistan	19103	440	4817	13846	0.023033
24	Chile	18435	247	9572	8616	0.013398
25	Singapore	17548	17	1347	16184	0.000969
26	Israel	16185	229	9593	6363	0.014149

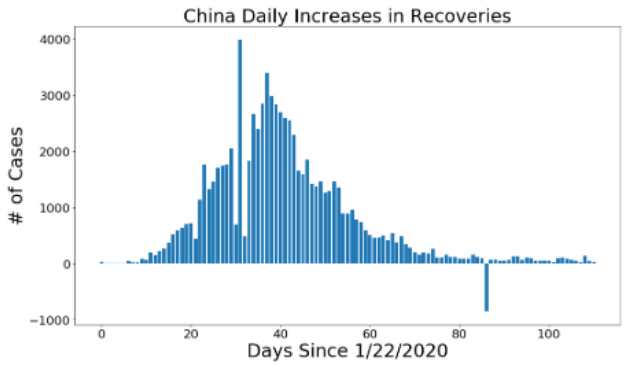
- **China cases**

```
country_plot(adjusted_dates, china_cases, china_daily_increase, china_daily_death, china_da
```



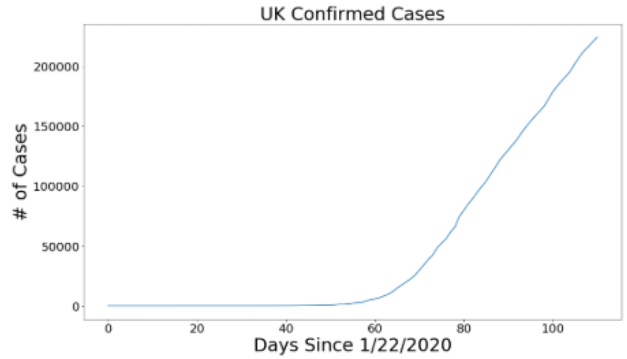


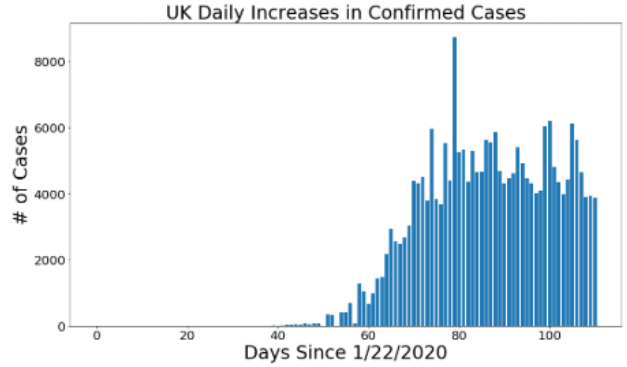


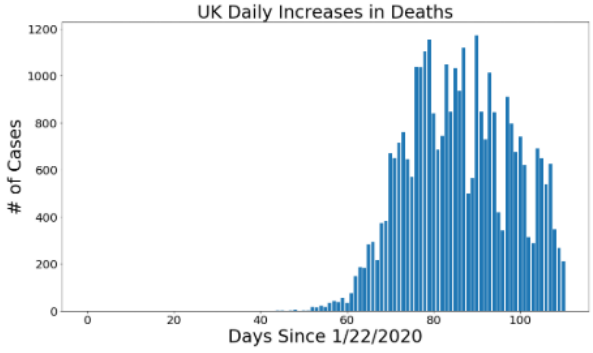


- **UK cases**

```
country_plot(adjusted_dates, uk_cases, uk_daily_increase, uk_daily_death, uk_d
```

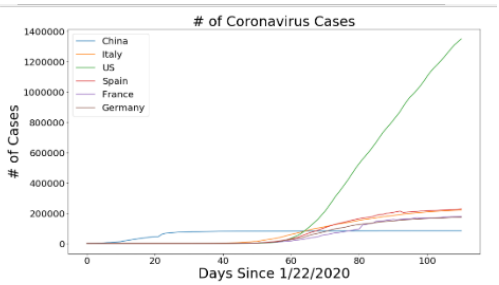


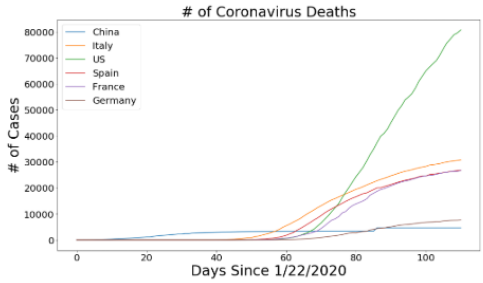


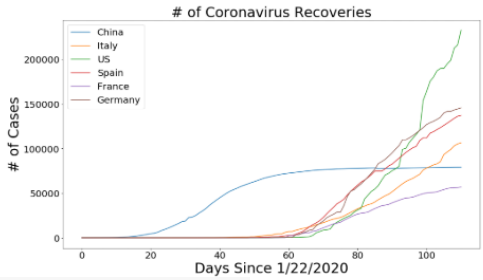


- **Country comparison**

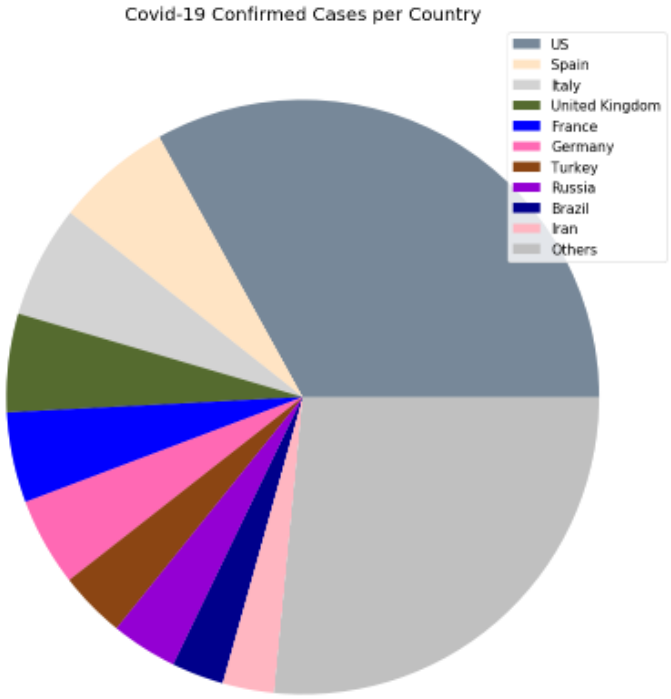
```
plt.figure(figsize=(16, 9))
plt.plot(adjusted_dates, china_cases)
plt.plot(adjusted_dates, italy_cases)
plt.plot(adjusted_dates, us_cases)
plt.plot(adjusted_dates, spain_cases)
plt.plot(adjusted_dates, france_cases)
plt.plot(adjusted_dates, germany_cases)
plt.title('# of Coronavirus Cases', size=30)
plt.xlabel('Days Since 1/22/2020', size=30)
plt.ylabel('# of Cases', size=30)
plt.legend(['China', 'Italy', 'US', 'Spain', 'France', 'Germany'], prop={'size
plt.xticks(size=20)
plt.yticks(size=20)
plt.show()

plt.figure(figsize=(16, 9))
plt.plot(adjusted_dates, china_deaths)
plt.plot(adjusted_dates, italy_deaths)
plt.plot(adjusted_dates, us_deaths)
plt.plot(adjusted_dates, spain_deaths)
plt.plot(adjusted_dates, france_deaths)
plt.plot(adjusted_dates, germany_deaths)

plt.title('# of Coronavirus Deaths', size=30)
plt.xlabel('Days Since 1/22/2020', size=30)
plt.ylabel('# of Cases', size=30)
plt.legend(['China', 'Italy', 'US', 'Spain', 'France', 'Germany'], prop={'size
plt.xticks(size=20)
plt.yticks(size=20)
plt.show()

plt.figure(figsize=(16, 9))
plt.plot(adjusted_dates, china_recoveries)
plt.plot(adjusted_dates, italy_recoveries)
plt.plot(adjusted_dates, us_recoveries)
plt.plot(adjusted_dates, spain_recoveries)
plt.plot(adjusted_dates, france_recoveries)
plt.plot(adjusted_dates, germany_recoveries)
plt.title('# of Coronavirus Recoveries', size=30)
plt.xlabel('Days Since 1/22/2020', size=30)
plt.ylabel('# of Cases', size=30)
plt.legend(['China', 'Italy', 'US', 'Spain', 'France', 'Germany'], prop={'size
plt.xticks(size=20)
plt.yticks(size=20)
plt.show()
```








```
c = random.choices(list(mcolors.CSS4_COLORS.values()),k = len(unique_countr
plt.figure(figsize=(20,15))
plt.title('Covid-19 Confirmed Cases per Country', size=20)
plt.pie(visual_confirmed_cases, colors=c)
plt.legend(visual_unique_countries, loc='best', fontsize=15)
plt.show()
```



- **Modeling**
- **Convert date format**

```
start = '1/22/2020'
start_date = datetime.datetime.strptime(start, '%m/%d/%Y')
future_forcast_dates = []
for i in range(len(future_forcast)):
    future_forcast_dates.append((start_date + datetime.timedelta(days=i)).strf
```

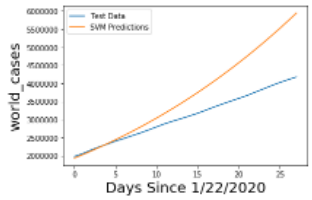
- **train_test_split**

```
d = train_test_split(days_since_1_22, world_cases, test_size=
```

- **check against testing data**

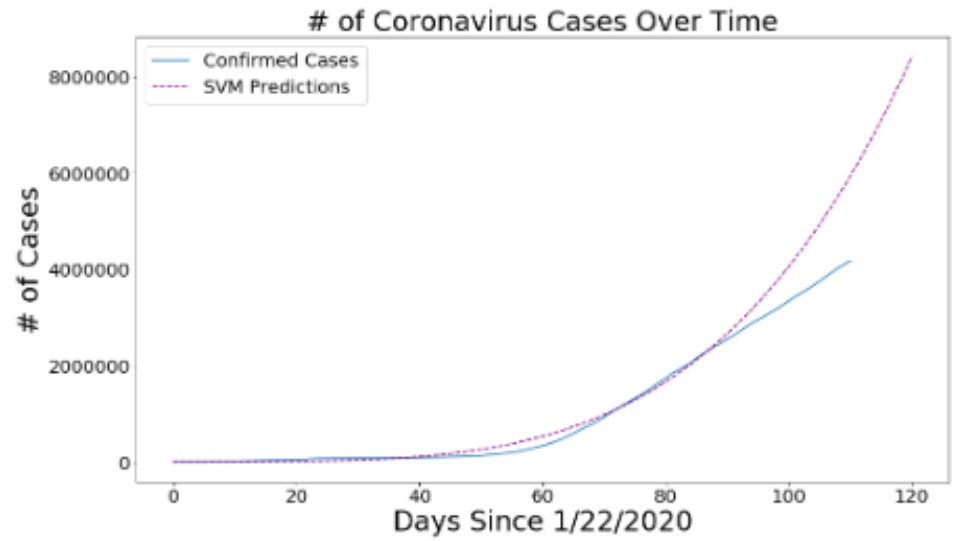
```
# check against testing data
svm_test_pred = svm_confirmed.predict(X_test_confirmed)
plt.plot(y_test_confirmed)
plt.plot(svm_test_pred)
plt.legend(['Test Data', 'SVM Predictions'])
plt.xlabel('Days Since 1/22/2020', size=20)
plt.ylabel('world_cases', size=20)
print('MAE:', mean_absolute_error(svm_test_pred, y_test_confirmed))#MAE (Mean .
print('MSE:',mean_squared_error(svm_test_pred, y_test_confirmed))#MSE (Mean Sq
print('R Square:',r2_score(y_test_confirmed, svm_test_pred))
```

```
MAE: 606416.1992438835
MSE: 664312834869.9006
R Square: -0.5577305593969273
```



- **Plot prediction**

```
plot_predictions(adjusted_dates, world_cases, svm_pred, 'SVM Predictions', 'p
```



```
# Future predictions using SVM
svm_df = pd.DataFrame({'Date': future_forcast_dates[-10:], 'SVM Predicted #
svm_df
```

	Date	SVM Predicted # of Confirmed Cases Worldwide
0	05/12/2020	6162348.0
1	05/13/2020	6387212.0
2	05/14/2020	6618180.0
3	05/15/2020	6855363.0
4	05/16/2020	7098870.0
5	05/17/2020	7348813.0
6	05/18/2020	7605304.0
7	05/19/2020	7868456.0
8	05/20/2020	8138385.0
9	05/21/2020	8415205.0

- **transform our data for polynomial regression**

```
# transform our data for polynomial regression
poly = PolynomialFeatures(degree=3)
poly_X_train_confirmed = poly.fit_transform(X_train_confirmed)
poly_X_test_confirmed = poly.fit_transform(X_test_confirmed)
poly_future_forcast = poly.fit_transform(future_forcast)

bayesian_poly = PolynomialFeatures(degree=4)
bayesian_poly_X_train_confirmed = bayesian_poly.fit_transform(X_train_confirme
bayesian_poly_X_test_confirmed = bayesian_poly.fit_transform(X_test_confirmed)
bayesian_poly_future_forcast = bayesian_poly.fit_transform(future_forcast)
```

```
# polynomial regression
linear_model = LinearRegression(normalize=True, fit_intercept=False)
linear_model.fit(poly_X_train_confirmed, y_train_confirmed)
test_linear_pred = linear_model.predict(poly_X_test_confirmed)
linear_pred = linear_model.predict(poly_future_forcast)
```

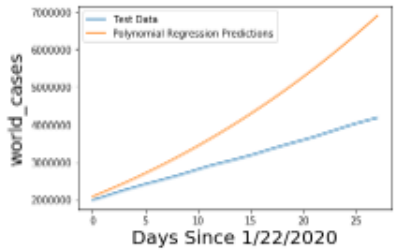
- **check against testing data**

```
print(linear_model.coef_)
```

```
[[-9.09392611e+04  2.34275159e+04 -9.96942331e+02  1.23820199e+01]]
```

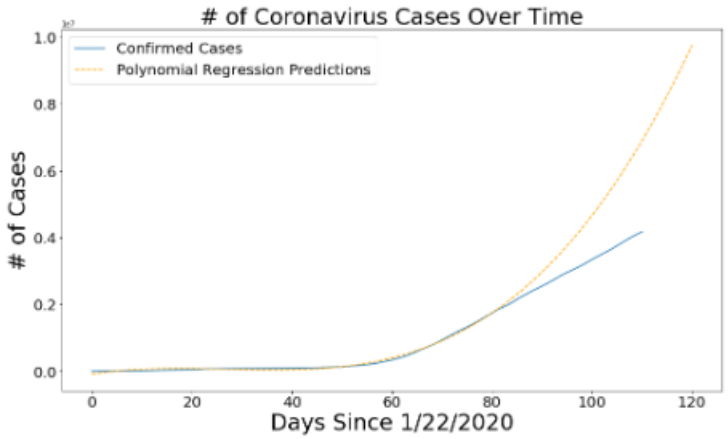
```
plt.plot(y_test_confirmed)
plt.plot(test_linear_pred)
plt.legend(['Test Data', 'Polynomial Regression Predictions'])
plt.xlabel('Days Since 1/22/2020', size=20)
plt.ylabel('world_cases', size=20)
print('MAE:', mean_absolute_error(test_linear_pred, y_test_confirmed))
print('MSE:',mean_squared_error(test_linear_pred, y_test_confirmed))
print('R Square:',r2_score(y_test_confirmed, svm_test_pred))
```

```
MAE: 1108730.6730235254
MSE: 1871084639793.898
R Square: -0.5577305593969273
```



- **Plot prediction**

```
]: plot_predictions(adjusted_dates, world_cases, linear_pred, 'Polynomial Regress
```



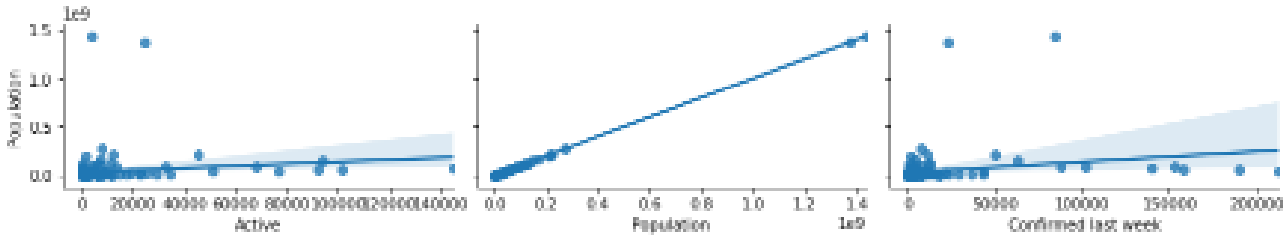

```
]: # Future predictions using polynomial regression
linear_pred = linear_pred.reshape(1,-1)[0]
svm_df = pd.DataFrame({'Date': future_forcast_dates[-10:], 'Polynomial Predict
svm_df
```

]:

	Date	Polynomial Predicted # of Confirmed Cases Worldwide
0	05/12/2020	7160223.0
1	05/13/2020	7423144.0
2	05/14/2020	7692393.0
3	05/15/2020	7968042.0
4	05/16/2020	8250167.0
5	05/17/2020	8538842.0
6	05/18/2020	8834141.0
7	05/19/2020	9136137.0
8	05/20/2020	9444907.0
9	05/21/2020	9760523.0

- **linear regression**
- **Plot pairwise relationships in a dataset**

```
sns.pairplot(country_wise_pop_weekly_latest, x_vars=['Active', 'Population', '
plt.show()
```



- **Collate feature column and Check data**

```
feature_cols = ['Active', 'Population', 'Confirmed last week']
X = country_wise_pop_weekly_latest[feature_cols]
print(type(X))
print(X.shape)
y = country_wise_pop_weekly_latest['New cases']
# 检查y
print(y.head())
```

```
<class 'pandas.core.frame.DataFrame'>
(185, 3)
0    232
1      7
2    158
3      2
4      0
Name: New cases, dtype: int64
```

```
X.isna().sum()
```

```
Active                 0
Population            12
Confirmed last week    0
dtype: int64
```

```
y.isna().sum()
```

```
0
```

```
X=X.fillna(X.mean())
```

```
X.isna().sum()
```

```
Active                0
Population            0
Confirmed last week   0
dtype: int64
```

```
print(np.isfinite(y).all())
```

```
True
```

```
print(np.isinf(y).all())
```

```
False
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1)
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(138, 3)
(138,)
(47, 3)
(47,)
```

```
linreg = LinearRegression()
model = linreg.fit(X_train, y_train)
# Output the constant terms of the model
print(linreg.intercept_)
# Output model coefficients
print(linreg.coef_)
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False
```

```
53.73622797586705
[ 4.42930593e-02  1.00053981e-06 -9.38831393e-03]
```

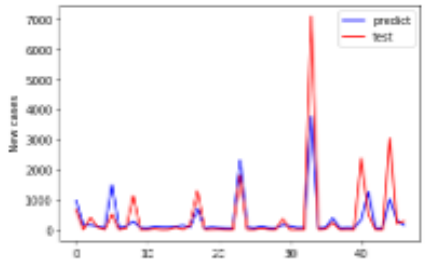
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=Fals
e)
```

- **Output the coefficient matrix of the characteristic variable and predicted results**

```
#Output the coefficient matrix of the characteristic variable
print(list(zip(feature_cols, linreg.coef_)))
# Output predicted results
y_pred = linreg.predict(X_test)
print(y_pred)
```

```
[('Active', 0.044293059250275164), ('Population', 1.0005398121306546e-06),
('Confirmed last week', -0.009388313933364518)]
[ 963.48728025  150.69168108  155.57588284  112.4603503    55.89722305
 1504.1437114    61.80036031  110.34931814  281.55328447   67.85069461
   56.47301585  101.59587142   86.96120024   84.36358221  105.37407862
  146.07465295   77.68260308  705.82023982   54.87221102   85.54380134
   65.17532053   55.4461459    52.43224606 2306.48435068   84.14462065
   54.86043328  116.27405744   58.65943057   54.52937848  158.85792107
  123.62534615   66.41235444   73.65702183 3765.35909394   61.65705278
   69.65421002  391.43557379   54.72860788   67.31941302   60.42206204
  331.86284948 1264.93743161   54.27362357   54.05165581 1020.64263584
  318.56479576  144.39008879]
```

```
plt.figure()
plt.plot(range(len(y_pred)),y_pred,'b',label="predict")
# Draw the true value of the test set
plt.plot(range(len(y_pred)),y_test,'r',label="test")
# Location of legend: upper right corner
plt.legend(loc="upper right")
plt.ylabel('New cases')
plt.show()
print('MAE:', mean_absolute_error(y_pred, y_test))#MAE (Mean Absolute Error) 
print('MSE:',mean_squared_error(y_pred, y_test))#MSE (Mean Square Error) 均方
print('R Square:',r2_score(y_pred, y_test))
```




```
MAE: 308.67956522904433
MSE: 482949.4033206914
R Square: -0.07109243643252894
```

- **Linear model versus the linear model**

```
SCORE = pd.DataFrame({'tool':['SupportVectorMachines(SVM)','PolynomialRegressi
                              'MAE:':[mean_absolute_error(svm_test_pred, y_test_confi
                              'MSE:':[mean_squared_error(svm_test_pred, y_test_confirm
                              'R Square:':[r2_score(svm_test_pred, y_test_confirmed),r
SCORE
```

	tool	MAE:	MSE:	R Square:
0	SupportVectorMachines(SVM)	6.064162e+05	6.643128e+11	0.538413
1	PolynomialRegression	1.108731e+06	1.871085e+12	0.108595
2	LinearRegression	3.086796e+02	4.829494e+05	-0.071092

- **LSTM**
- **convert an array of world_cases into a dataset matrix**

```
dataset_LSTM = world_cases.astype('float32')
def create_dataset_LSTM(dataset_LSTM, look_back = 3):
    data_LSTMX, data_LSTMY = [], []
    for i in range(len(dataset_LSTM)-look_back-1):
        a = dataset_LSTM[i:(i+look_back),0]
        data_LSTMX.append(a)
        data_LSTMY.append(dataset_LSTM[i + look_back, 0])
    return np.array(data_LSTMX), np.array(data_LSTMY)
```

- **fix random seed for reproducibility and normalize the dataset**

```
# fix random seed for reproducibility
np.random.seed(7)

# normalise the dataset
scaler = MinMaxScaler(feature_range=(0,1))
dataset_LSTM = scaler.fit_transform(dataset_LSTM)
```

- **split into train and test sets**

```
# split into train and test sets
train_size = int(len(dataset_LSTM)*0.3)
test_size = len(dataset_LSTM) - train_size
train, test = dataset_LSTM[0:train_size,:], dataset_LSTM[train_siz
```

- **use this function to prepare the train and test datasets for modeling**

```
# use this function to prepare the train and test datasets for mod
look_back = 1
trainX, trainY = create_dataset_LSTM(train, look_back)
testX, testY = create_dataset_LSTM(test, look_back)
```

- **reshape input to be [samples, time steps, features] and create and fit the LSTM network**

```
# reshape input to be [samples, time steps, features]
trainX = np.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1]))
testX = np.reshape(testX, (testX.shape[0], 1, testX.shape[1]))

# create and fit the LSTM network
model = Sequential()
model.add(LSTM(4, input_shape=(1, look_back)))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(trainX, trainY, epochs=100, batch_size=1, verbose=2)
```

```
Epoch 1/100
 - 1s - loss: 5.4880e-05
Epoch 2/100
 - 0s - loss: 4.4793e-05
Epoch 3/100
 - 0s - loss: 4.3471e-05
Epoch 4/100
 - 0s - loss: 4.3654e-05
Epoch 5/100
 - 0s - loss: 4.1579e-05
Epoch 6/100
 - 0s - loss: 4.0793e-05
Epoch 7/100
 - 0s - loss: 3.7429e-05
Epoch 8/100
 - 0s - loss: 4.1110e-05
Epoch 9/100
 - 0s - loss: 3.6098e-05
Epoch 10/100
 - 0s - loss: 3.4766e-05
```

- **make predictions and invert predictions**

```
# make predictions
trainPredict = model.predict(trainX)
testPredict = model.predict(testX)
```

```
# invert predictions
trainPredict = scaler.inverse_transform(trainPredict)
trainY = scaler.inverse_transform([trainY])
testPredict = scaler.inverse_transform(testPredict)
testY = scaler.inverse_transform([testY])
```

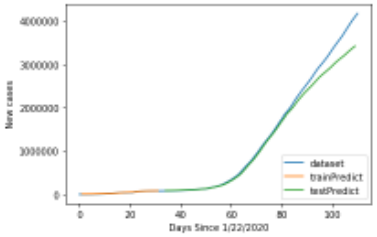
- **calculate MSE**

```
# calculate MSE
trainScore = math.sqrt(mean_squared_error(trainY[0], trainPredict[:,0]))
print('Train Score: %.2f RMSE' % (trainScore))
testScore = math.sqrt(mean_squared_error(testY[0], testPredict[:,0]))
print('Test Score: %.2f RMSE' % (testScore))
```

```
Train Score: 4435.85 RMSE
Test Score: 215223.13 RMSE
```

- **plot baseline and predictions**

```
# shift test predictions for plotting
testPredictPlot = np.empty_like(dataset_LSTM)
testPredictPlot[:, :] = np.nan
testPredictPlot[len(trainPredict)+(look_back*2)+1:len(dataset_LSTM)-1, :] =

# plot baseline and predictions
plt.plot(scaler.inverse_transform(dataset_LSTM), label = 'dataset')
plt.plot(trainPredictPlot, label = 'trainPredict')
plt.plot(testPredictPlot, label = 'testPredict')
plt.legend(loc="lower right")
plt.xlabel('Days Since 1/22/2020')
plt.ylabel('New cases')
plt.show()
```



End of Python code