I. Abstract:

This project focusses on prediction of cumulative number of confirmed COVID19 cases in various locations across the world. Additionally, the total number of fatalities for future dates is also an important parameter in this prediction problem. Since this is a quantitative output value hence it is a regression problem (RMSLE being our prime performance metric). We utilize 3 different methods i.e. 1) Linear Regression, 2) Polynomial Regression and 3) XG Boost Decision Tree along with Regression. Additionally, optimal fine tuning of parameters is carried out for different models which yields us our best performing model.

II. Methodology:

1) Import all necessary libraries

- For our implementation we utilize necessary libraries such as NumPy, Seaborn, Matplotlib & Pandas for data visualization & data pre-processing

2) Import the dataset

- We obtain the data from Kaggle's competition page. The dataset essentially consists of 2 .csv files (train.csv and test.csv) with several data attributes along with 1 submission.csv file to be submitted on Kaggle.

3) Explore & Visualize the data to get an intuition of various attributes.

- To get a proper intuition of our data we apply several built-in methods from Python to get the dimensions and label size for both the training as well as test dataset.
  Data visualization in this project is critical since it lets us have a clear picture about the impact caused by the COVID virus in different geographical regions along with several other attributes. Once we have a clear depiction of all the different attributes related to the days, dates and geographical region, it will help us achieve an optimized model.

4) Data Preprocessing

- This step involves handling of missing values of several entities such as Nan's for example (Not a Number entities). This step is critical as we need to make sure that our dataset is free from redundant entries which will not contribute to the model building stage.

- Data Merging is also an important step in pre-processing; but here all our features are already present in the training.csv hence this is not needed. However, we need to create new attributes such as Day Counts or new Confirmed cases or New fatalities. This is carried out in the pre-processing step is usually the most critical stage of any ML Pipeline

5) **Model A -Linear Regression**

- 2 Separate Linear Regression Models are developed for Confirmed cases and Fatalities and error metrics such as RMSLE and MSE are calculated.

6) **Model B - Polynomial Regression**

- As Polynomial Regression fits the curve better compared to a Linear Regression Model, we experiment with various value of n (degree of the polynomial) such as 2,3,5.

7) **Model C -XGB Regression**

- This is the best performing model as it uses decision tree with the combination of a regression model so that the model scales well on large amounts of data. Additionally, fine tuning of hyperparameters is also carried out in this step such as Learning Rate, subsample rate, n_estimators, L1/L2 Regularization, max Delta, seed etc.

8) **Performance Evaluation and Observations**

- This involves a thorough discussion about which parameters are critical during model building stage and the effect of various techniques on performance metrics.

III. Dataset Description:

- This project mainly deals with 3 .csv files i.e. train, test and submission.

- The train.csv is essentially the training data which has several features such as Province State, Country Region, Date, Confirmed Cases and Fatalities. Size for the train.csv is **(35369 x 6)**. One important thing to note is that since this is Week 4 of the Competition, it has data merged from previous datasets since data entities from January 2020 are present.

- The test.csv file is used to predict the date and it has attributes such as Forecast ID, Province State, Country Region and Date. The size for this is **(13459 x 4)**

- Finally, the submission.csv file is used for submissions on Kaggle. The attributes for this are Forecast ID, number of Confirmed Cases and Fatalities.

- The size for train and test can be shown below:

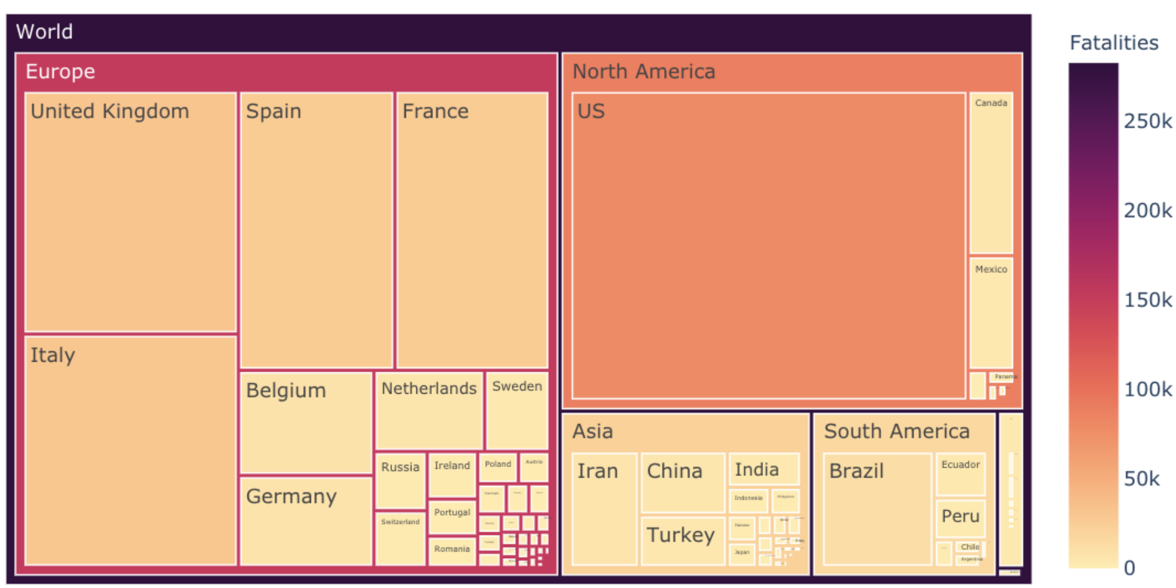
```
(35995, 5)

(13459, 5)
```

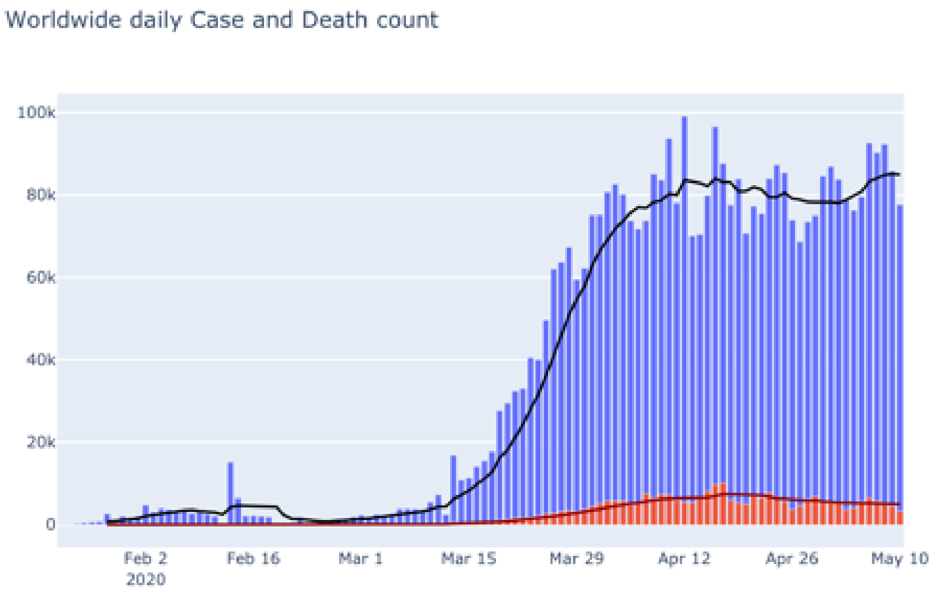
IV. <u>Data Visualization:</u>

1) Current share of COVID-19 deaths can be displayed as follows:

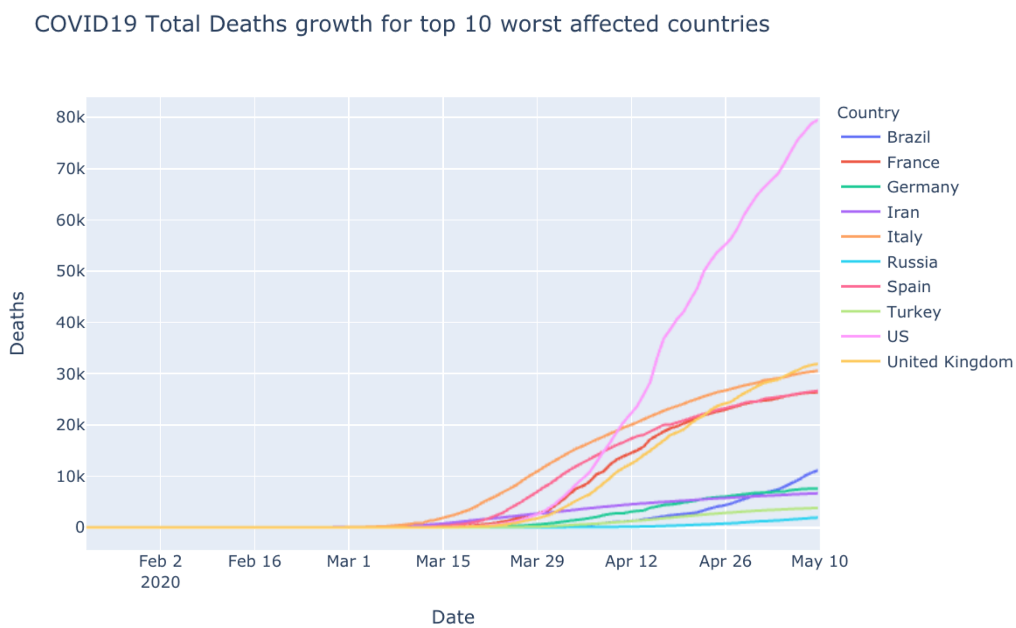
Current share of Worldwide COVID19 Deaths



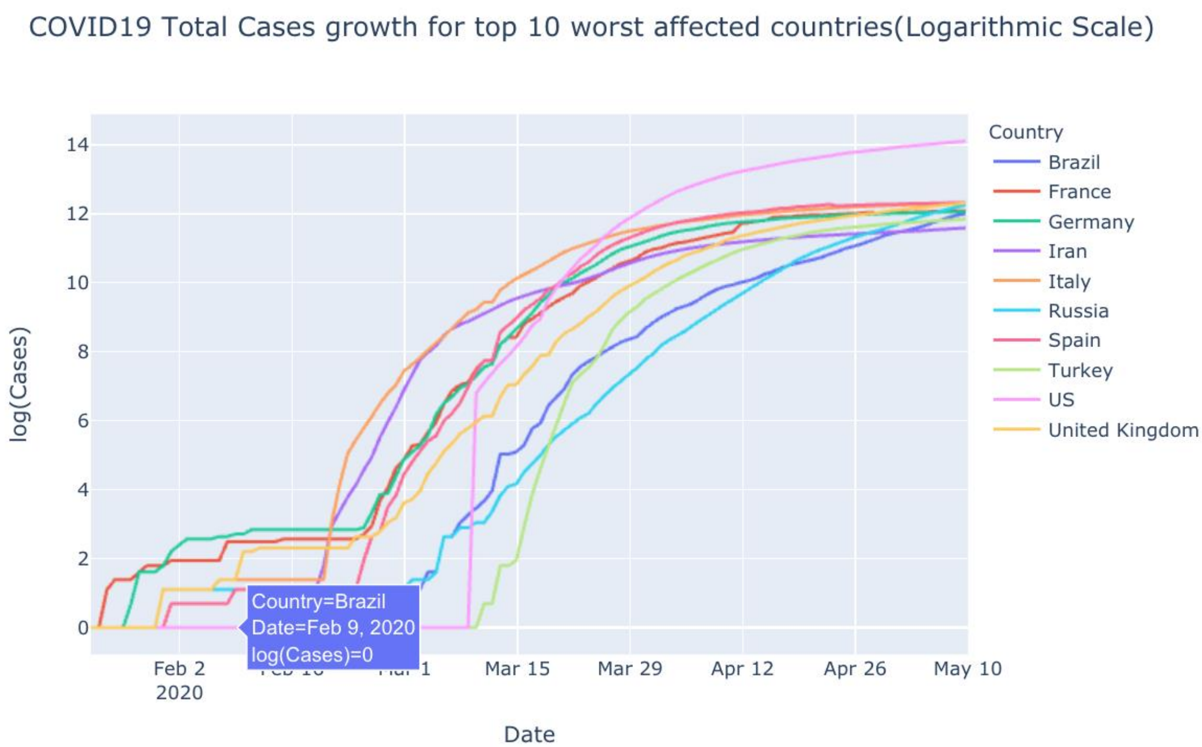
2) Worldwide death case and death count can be explained as follows:



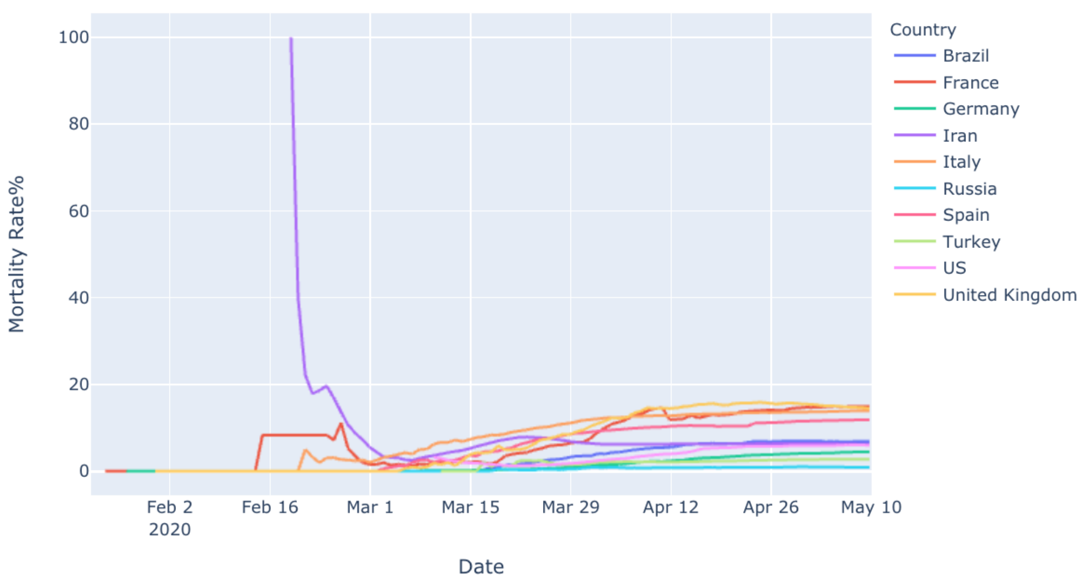
3) The total deaths for top 10 worst affected countries can be given as follows:



4) The same setting on logarithmic scale is displayed below:



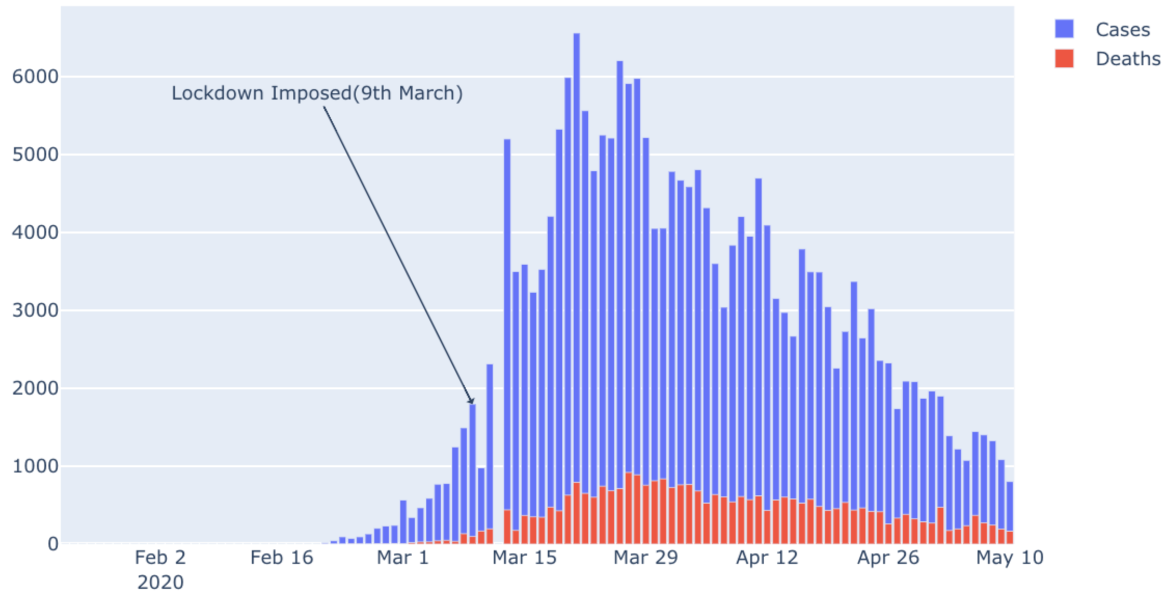
5) The mortality rate for top 10 worst affected countries is given below:



6) State wise breakdown of Cases for India is given below:

	State/UnionTerritory	Cured	Deaths	Confirmed
0	Maharashtra	4786	868	23401
1	Gujarat	2780	513	8541
2	Tamil Nadu	2051	53	8002
3	Delhi	2129	73	7233
4	Rajasthan	2264	113	3988
5	Madhya Pradesh	1747	221	3785
6	Uttar Pradesh	1758	80	3573
7	West Bengal	499	190	2063
8	Andhra Pradesh	975	45	2018
9	Punjab	168	31	1877
10	Telangana	800	30	1275
11	Jammu and Kashmir	427	10	879
12	Karnataka	426	31	862
13	Bihar	377	6	747
14	Haryana	337	11	730
15	Kerala	489	4	519
16	Orissa	85	3	414
17	Chandigarh	24	2	174
18	Jharkhand	78	3	160
19	Tripura	2	0	152
20	Uttarakhand	46	1	68
21	Assam	34	2	65
22	Chhattisgarh	53	0	59
23	Himachal Pradesh	39	2	59
24	Ladakh	21	0	42
25	Andaman and Nicobar Islands	33	0	33
26	Meghalaya	10	1	13
27	Puducherry	6	0	12
28	Goa	7	0	7
29	Manipur	2	0	2
30	Dadar Nagar Haveli	0	0	1
31	Mizoram	1	0	1
32	Arunachal Pradesh	1	0	1

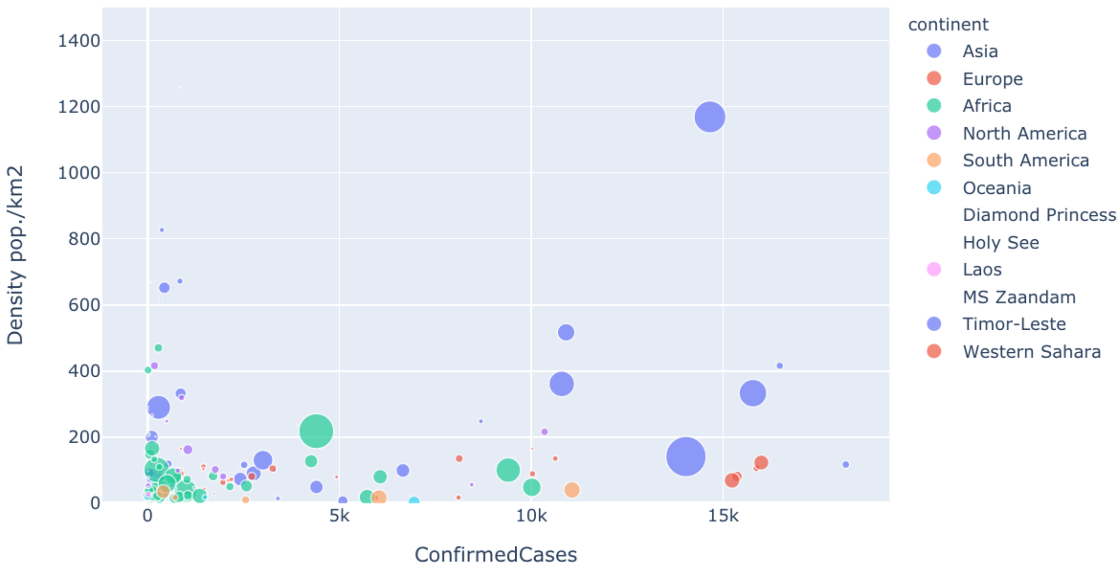
7) Daily Death case in Italy is given as follows:



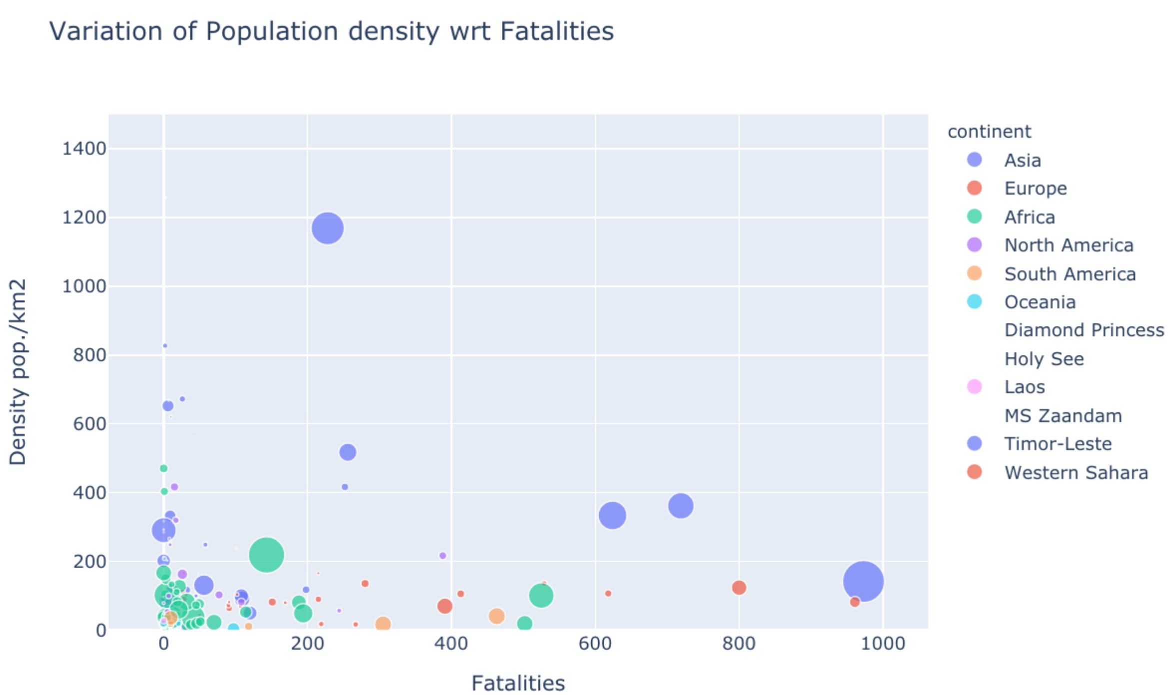
8) Region wise data for Italy is as follows:

	RegionName	TotalHospitalizedPatients	Recovered	Deaths	TotalPositiveCases	TestsPerformed
0	Lombardia	5738	36406	15054	81871	292603
1	Piemonte	2156	12038	3400	28776	147318
2	Emilia-Romagna	1678	15969	3867	26876	151040
3	Veneto	438	11615	1666	18741	250175
4	Toscana	424	4764	950	9787	132464
5	Liguria	522	4695	1293	8832	41535
6	Lazio	1349	2334	562	7190	143970
7	Marche	305	2352	964	6543	50206
8	Campania	459	2301	392	4602	54822
9	Puglia	372	1332	451	4327	55794
10	P.A. Trento	95	3119	443	4297	31970
11	Sicilia	287	1020	257	3339	92609
12	Friuli Venezia Giulia	99	1996	312	3138	57130
13	Abruzzo	240	1132	366	3107	34428
14	P.A. Bolzano	70	1835	290	2572	22500
15	Umbria	44	1233	71	1412	33027
16	Sardegna	94	712	120	1343	30582
17	Valle d'Aosta	45	912	139	1158	7651
18	Calabria	65	473	93	1134	45438
19	Basilicata	47	217	27	386	17774
20	Molise	11	132	22	383	9247

9) The variation of population density with respect to confirmed cases



<u>For Fatalities:</u>



V. Data Pre-processing:

1) Addition of new features such as Month and Day into existing data. The output obtained is as follows:

	Id	Province_State	Country_Region	Date	ConfirmedCases	Fatalities	Month	Day
0	1	NaN	Afghanistan	2020-01-22	0.0	0.0	1	22
1	2	NaN	Afghanistan	2020-01-23	0.0	0.0	1	23
2	3	NaN	Afghanistan	2020-01-24	0.0	0.0	1	24
3	4	NaN	Afghanistan	2020-01-25	0.0	0.0	1	25
4	5	NaN	Afghanistan	2020-01-26	0.0	0.0	1	26

2) Calculation of number of unique country regions and province states. It can be shown as below:

For train

```
Datatrain
Number of Country_Region:  184
Number of Province_State:  133
Number of Days:  115
Number of datapoints in train: 35995
L Trains: 313
```

For Test

```
Datatest
Number of Days:  43
Number of datapoints in test: 13459
L Test: 313
```

3) Changing the Date format to yy/dd/mm. The result is displayed as follows:

```
0        2020-01-22
1        2020-01-23
2        2020-01-24
3        2020-01-25
4        2020-01-26
            ...
35364    2020-05-09
35365    2020-05-10
35366    2020-05-11
35367    2020-05-12
35368    2020-05-13
Name: Date, Length: 35369, dtype: datetime64[ns]
```

4) Filling empty data entities with 0. This is done using fillna () method in Python.
5) For Polynomial Regression we follow a different path for feature engineering. This can be explained as follows:

    a) Combining Country and Province: The result is as follows:

	Id	Province_State	Country_Region	Date	ConfirmedCases	Fatalities	Region
0	1		Afghanistan	2020-01-22	0.0	0.0	Afghanistan
1	2		Afghanistan	2020-01-23	0.0	0.0	Afghanistan
2	3		Afghanistan	2020-01-24	0.0	0.0	Afghanistan
3	4		Afghanistan	2020-01-25	0.0	0.0	Afghanistan
4	5		Afghanistan	2020-01-26	0.0	0.0	Afghanistan
...	...	...	...	...	...	...	...
35364	35677		Zimbabwe	2020-05-09	35.0	4.0	Zimbabwe
35365	35678		Zimbabwe	2020-05-10	36.0	4.0	Zimbabwe
35366	35679		Zimbabwe	2020-05-11	36.0	4.0	Zimbabwe
35367	35680		Zimbabwe	2020-05-12	36.0	4.0	Zimbabwe
35368	35681		Zimbabwe	2020-05-13	37.0	4.0	Zimbabwe

b) Adding a new feature which estimates the number of days since the first case was found. The result obtained is displayed below:

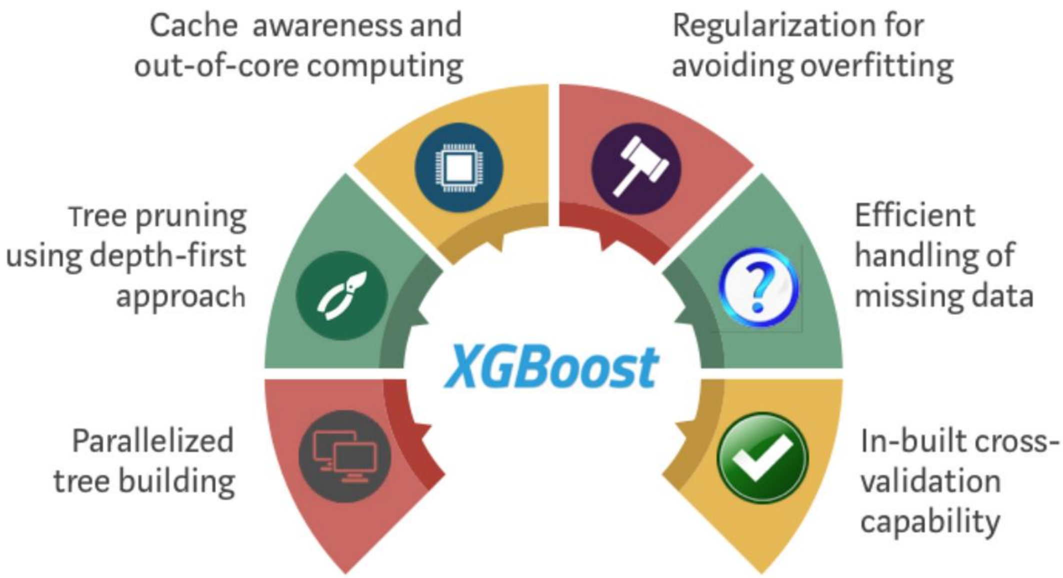
Province_State	Country_Region	Date	ConfirmedCases	Fatalities	Region	days_since_p0_world
	Afghanistan	2020-01-22	0.0	0.0	Afghanistan	0
	Afghanistan	2020-01-23	0.0	0.0	Afghanistan	1
	Afghanistan	2020-01-24	0.0	0.0	Afghanistan	2
	Afghanistan	2020-01-25	0.0	0.0	Afghanistan	3
	Afghanistan	2020-01-26	0.0	0.0	Afghanistan	4
...	...	...	...	...	...	...
	Zimbabwe	2020-05-09	35.0	4.0	Zimbabwe	108
	Zimbabwe	2020-05-10	36.0	4.0	Zimbabwe	109
	Zimbabwe	2020-05-11	36.0	4.0	Zimbabwe	110
	Zimbabwe	2020-05-12	36.0	4.0	Zimbabwe	111
	Zimbabwe	2020-05-13	37.0	4.0	Zimbabwe	112

c) Adding new features further such as number of days since the first case was detected in a particular country as well as in a particular region. It is evident from the below chart.

Date	ConfirmedCases	Fatalities	Region	days_since_p0_world	days_since_p0_country	days_since_p0_region
2020-01-23	9.0	0.0	China Anhui	1	1	1
2020-01-23	22.0	0.0	China Beijing	1	1	1
2020-01-23	9.0	0.0	China Chongqing	1	1	1
2020-01-23	5.0	0.0	China Fujian	1	1	1
2020-01-23	2.0	0.0	China Gansu	1	1	0
2020-01-23	32.0	0.0	China Guangdong	1	1	1
2020-01-23	5.0	0.0	China Guangxi	1	1	1
2020-01-23	3.0	0.0	China Guizhou	1	1	1
2020-01-23	5.0	0.0	China Hainan	1	1	1
2020-01-23	1.0	1.0	China Hebei	1	1	1
2020-01-23	2.0	0.0	China Heilongjiang	1	1	0

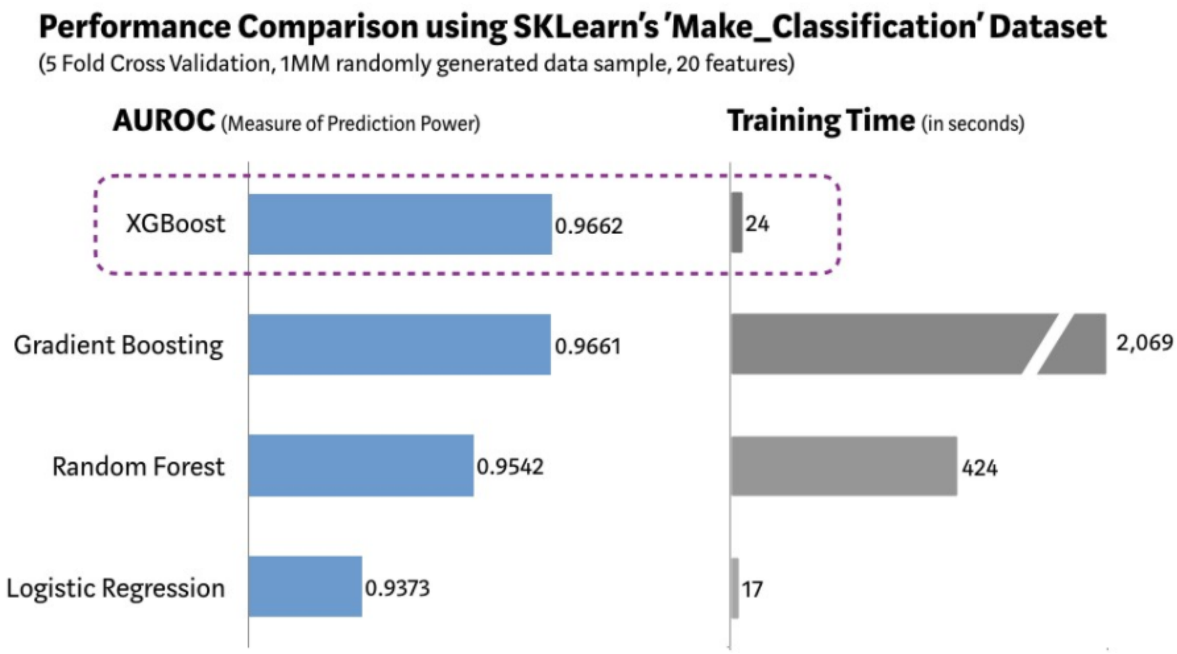
VI. <u>Advantages of XG Boost Decision Tree Algorithm (Model 3)</u>

- XGBoost boost is a machine learning algorithm that can be used for regression as well as classification problems as well as user defined prediction problems
- It is a decision tree implementation that is fairly easy to visualize and develop interpretable models.



How XGBoost optimizes standard GBM algorithm

- Additionally, XGB also allows several system level optimizations such as Parallelization, Tree Pruning as well as hardware optimizations.
- On the algorithmic side, there are several enhancement techniques such as Regularization, Cross Validation, Weighted Quantile search as well as sparsity awareness.
- Thus, to sum it up, XG Boost offers the perfect combination of prediction performance along with processing time required as compared to other algorithms
- One important thing to consider is the special focus on hyperparameter tuning to obtain the best performing model overall.



XGBoost vs. Other ML Algorithms using SKLearn's Make_Classification Dataset

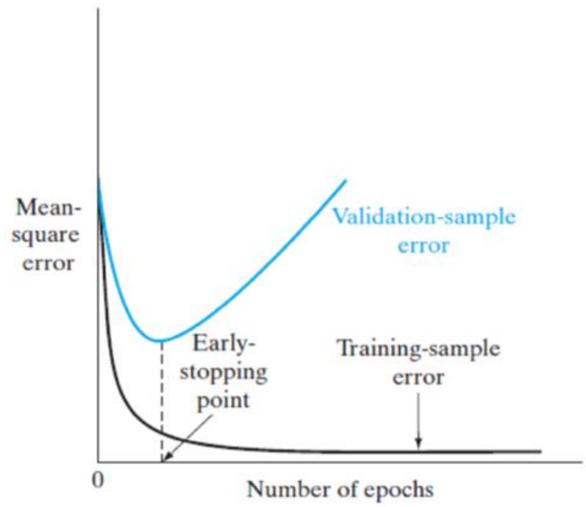
VII. Key Considerations while developing any Machine Learning Model:

- **Cross validation**

Cross validation technique is necessary to avoid the overfitting to the data while training the model. It assures the good generalization and improve performance of our model. Different cross validation schemes such as K-Fold Cross Validation, LOOCV (Leave One out cross Validation) or Validation set approach can be used.

- **Early Stopping**

When we start training the model both training, and the test errors start decreasing up to a certain point. If we continue to train our model it will start over fitting to the training data, so the training error continues to decrease but the generalization error starts increasing due to overfitting. So, to prevent this we should stop training the model after this point called the early stopping point. This can be illustrated from the figure below



- **Confusion Matrix**

Confusion Matrix gives us insight not only into the misclassifications being made by a classifier but also the types of errors that are being made. Thus, A confusion matrix consists of summary of probable predictions. Thus, confusion matrix is mainly used for error analysis to find False positives and True Positives in our classification problem. Correct classification data is always available at the diagonal while non diagonal elements show misclassified data i.e. one class represented falsely into some other classification label. We use it for error analysis. We get correctly classification data at the diagonal.

- **One hot Encoding**

One hot encoding can be described as a methodology in which categorical variables are transformed into a form that can be fed to Machine Learning tasks to do a better job in prediction. If the output of our model is going to show the probabilities of where an image should be categorized as a prediction, each element represents the predicting probability of each classes.

VIII. Model Development

Model 1 Linear Regression Model

- We use 2 different linear Regression Models, one for number of Confirmed cases while the other for the number of Fatalities.
- Here we train the linear regression model with respect to
- The RMSLE for confirmed cases and fatalities can be given as:

```
RMSE for Confirmed Cases
Training - Mean Squared Log Error is:  28.291217359533853
Training - ROOT Mean Squared Log Error is:  5.318948896119783


(35369,)
```

RMSLE (Confirmed cases) = 5.31

```
RMSE for Fatalities
Training - Mean Squared Log Error is:  18.611034610097132
Training - ROOT Mean Squared Log Error is:  4.314050835363108


(35369,)
```

RMSLE(Fatalities) = 4.31

Some of the issues with Linear Regression can be explained as follows:

- Linear Regression is limited to linear relationships as it looks for the relationship between dependent and independent variables. This means that it assumes that there exists a straight-line relation. This is however not true in all cases.

- Linear Regression only focusses on the mean of the dependent variable as it looks at the interdependence between mean of the dependent and independent variables.
- Since mean is not a complete description of a single variable, in the same way Linear regression does not hold true regarding the relationship between different variables
- Additionally, Linear Regression is quite sensitive to the outliers in the data.
- For linear regression is work as expected, the data must be independent, but this is always not the case. An example where this does not hold true is clustering in time. Cases where we have to calculate the same attributes or features of each person multiple times. In such a scenario the data is not independent since one feature is inter dependent on the other.

  Thus, we move ahead with Polynomial Regression

Model 2: Polynomial Regression

- Here we implement Polynomial Regression with various degrees of the polynomial (values of n)
- One important thing to note here is that the input features need to be transformed to use with Polynomial Regression function (Feature Engineering done separately as compared to Linear Regression Model)

For n=1:

```
RMSLE for confirmed cases is 4.82547589930882
RMSLE for Fatalities is 3.5513218388997956
```

For n=2

```
RMSLE for confirmed cases is 4.827590111578429
RMSLE for Fatalities is 3.5482948182345364
```

For n=5

```
RMSLE for confirmed cases is 4.833270132578101
RMSLE for Fatalities is 3.547352016812334
```

For n=10

```
RMSLE for confirmed cases is 4.841759574055503
RMSLE for Fatalities is 3.5461873140058993
```

For n=25

```
RMSLE for confirmed cases is 4.835834136893844
RMSLE for Fatalities is 3.5417102654375223
```

For n=50

```
RMSLE for confirmed cases is 4.817507998298597
RMSLE for Fatalities is 3.5356295521655348
```

Selecting the best degree of Polynomial

- Looking at the above error metrics we select n=5 as the optimum value of degree for the polynomial. For higher values of n, the training time is extremely high, but the error does not reduce after a certain point, i.e. the data starts overfitting.
- Another way to solve this issue is using Bayesian Model selection. By implementing this model complexity and data fit can be easily obtained.

Reason Polynomial Regression performs better:

- The straight line in case of Linear Regression is unable to capture the patterns in the data which is also termed as underfitting. Depending upon on dataset which is fit for Polynomial Regression we can often observe that the RMSE decreases while the R^2 score increases as compared to the Linear Regression straight line.

Model 3: XG Boost Decision Tree

Default Configuration Settings:

1. N_estimators=1000
2. Gamma=0
3. Learning Rate =0.05
4. Random state= 42
5. Max Depth= 20

Varying the Learning rate:

It controls the learning in gradient boosting as it is applied by weighted factors for the addition of new trees into the model.
Step size shrinkage is used to prevent the issue of overfitting as it shrinks the feature weights to make the boosting process iterative in nature.

Learning Rate	0.05	0.5	0.999
RMSLE (Confirmed cases)	0.0012	0.001	0.009

Learning rate for 0.5 gives us the best results

Varying the col samples by tree

It randomly samples the feature for each tree. In a way this is done to limit or reduce the dimensionality prior to constructing trees.

Colsamples_bytree	0.3	0.5	0.8	0
RMSLE (Confirmed cases)	5.61	0.23	0.009	6.61

Col samples by tree for 0.8 gives us the best results.

Varying the max delta.

It is the maximum delta step we allow each output leaf to be. If value is 0 then there is no constraint.

max delta	0	0.5	-0.5
RMSLE (Confirmed cases)	0.009	0.009	0.009

Effect of L2 Regularization (lambda)

```
-------Confirmed Case---------s
Training - Mean Squared Error is:  3.5378302323470715e-07
Training - Mean Squared LOG Error is:  1.9214580000522764e-08
Training - ROOT Mean Squared LOG Error is:  0.00013861666566658846


(35995,)


--------Fatalities-------
Training - Mean Squared Error is:  0.0
Training - Mean Squared LOG Error is:  3.704744446624608e-15
Training - ROOT Mean Squared LOG Error is:  6.086661192003879e-08
```

RMSLE reduces when we use L2 Regularization. L2 regularization focusses term on weights thus making the model more conservative.

Varying the alpha:
RMSLE is worse compared to L2 regularization. L1 regularization focusses term on weights thus making the model more conservative.

```
-------Confirmed Case---------s
Training - Mean Squared Error is:  3.826024213907704e-06
Training - Mean Squared LOG Error is:  8.166507637498605e-07
Training - ROOT Mean Squared LOG Error is:  0.0009036873152533793


(35995,)


--------Fatalities-------
Training - Mean Squared Error is:  0.0
Training - Mean Squared LOG Error is:  3.704744446624608e-15
Training - ROOT Mean Squared LOG Error is:  6.086661192003879e-08
```

The results are as follows:

alpha	0	1
RMSLE (Confirmed cases)	0.009	0.043

Varying the n_estimators:

This is the number of trees to build for the decision tree. There needs to be a proper balance between the training time and the error metrics so as to prevent overfitting.

N_estimators	10	2000	4000
RMSLE (Confirmed cases)	0.366	0.0009	0.0009

Varying the sub sample rate:

This is the ratio that is considered for training instances. Making it half will lead to random sampling of half of the training data before growing trees.

Sub sample	0	0.25	0.5	0.99
RMSLE (Confirmed cases)	4.61	13.75	0.0002	0.0007

Varying the min child weight:

The is the minimum weight needed in a child node. If the tree partition occurs and leads to leaf node then the building process will give up further portioning.

Min_child weight	1	50
RMSLE (Confirmed cases)	0.0002	3.31

Here min-child weight =1 gives best results.

Varying the max depth:

It is the maximum depth of a tree. If we increase this parameter too much, it makes our model more complex and hence as a result it is most likely to overfit.

Max_depth	1	50	20
RMSLE (Confirmed cases)	5.3	0.0002	0.0002

Varying the seed:

This is the random number used in tree representation. Default value is 0.

seed	0	10	100
RMSLE (Confirmed cases)	0.0003	13.75	0.0002

IX. Results

Thus, various performance enhancements were observed after tweaking many parameters. Below represents our best Model Architecture Settings:
XG Boost combined with regression using below mentioned hyper parameter tuning gives us the best results

Best Configuration Settings (XGB):

1. n_estimators = 2000
2. gamma = 0
3. learning rate = 0.9999
4. random state = 42
5. max_depth = 20
6. max_delta=-0.5
7. reg_lambda=1
8. reg_alpha=0
9. subsample=0.5
10. min_child_weight=1
11. seed=10

Best RMSLE: 0.0002 (Confirmed cases) and 0.000006 (Fatalities)

```
-------Confirmed Case---------s
Training - Mean Squared Error is:  2.7817061276255973e-07
Training - Mean Squared LOG Error is:  8.521496203233144e-08
Training - ROOT Mean Squared LOG Error is:  0.00029191601880049586


(35995,)


--------Fatalities-------
Training - Mean Squared Error is:  0.0
Training - Mean Squared LOG Error is:  3.704744446624608e-15
Training - ROOT Mean Squared LOG Error is:  6.086661192003879e-08
```

X. Observations/Conclusion:

- While selecting between Linear Regression vs Polynomial regression it is extremely critical to consider the curve-linear relationship after which we can decide if polynomial is a better choice or not. This is usually done using univariate and bivariate inspections of data
- While selecting the value of n it is ideal to have the perfect spot between training time and our error performance metric (RMSLE in this case). We also need to be aware about overfitting as the value of n keeps on increasing.

- Problems involving huge amounts of data such as this case, it is always beneficial to go with a combination of regression combined with decision trees as they transform the data into a tree representation.
- In our implementation of decision trees, we utilized XG Boost as it offers the ideal sweet spot between high performance and accuracy as compared to other algorithms. As already discussed, before it offers many techniques such as regularization, parallel Processing, Handling of missing values, cross validation etc.
- In terms of parameter tuning which usually plays a critical role in any machine learning problem, factors such as learning rate, number of estimators/number of trees, lambda (L2 Regularization), sub sample and max depth may a critical role.

XI. Kaggle Rank:

Week 3 rank

88	Mehar Ganesh	0.08601	6	2d
89	EE257_S20_AA	0.08601	2	1m

Your Best Entry ↑

Your submission scored 0.63455, which is not an improvement of your best score. Keep trying!

90	Hrushikesh	0.08875	19	20h
91	huahua	0.08982	2	3d

Although, I had a Linear Regression model working with the week 3 dataset the competition was just 2 days away to end, hence I moved on with week 4.

Week 4 rank

200	▼ 144	Adityaraj Singh Chouhan	</> kernel202f681d26	1.03027	2	1mo
201	▼ 12	Tai-Hsien Ou Yang	</> COVID19_Week4	1.03157	1	1mo
202	▼ 11	EE257_S20_AA	</> Covid-19 Forecasti...	1.03160	2	1mo
203	▼ 45	Gaurav Rawat	</> Covid Week 4 Fore...	1.03272	5	1mo
204	▼ 141	mldlai	</> Covid19GlobalW4	1.03408	26	1mo

<u>Important:</u> Since this competition doesn't allow newer model evaluations after the competition ends (which is usually a short duration) hence my newer models such as Polynomial Regression and Decision Tree couldn't be evaluated for a better rank.

Hence the Rank is not a correct measure of performance since it still evaluates my Model 1- Linear Regression

XII. Issues/ Things that I wanted to try:

- Implement a Neural Network such as LSTM to predict the accurate future predictions. But since neural networks are out of scope for this class hence that's not a valid option.
- Try Light Gradient Boost Decision Tree Algorithm and compare it with my Model 3 based on XGB.