COVID-19 Global Forecasting

Talha Hanif Butt

The challenge involved forecasting confirmed cases and fatalities between April 1 and April 30 by region

Dataset

Available on Kaggle!

- 22,032 training records
- 13,158 test records
- Province_State, Country_Region, Date, Confirmed Cases, Fatalities
- Province_State, Country_Region, Date

Submissions were evaluated using root mean square logarithmic error as:

$$\sqrt{\frac{1}{n}\Sigma_{i=1}^{n}\left(log(p_i+1)-log(a_i+1)\right)^2}$$

where:

n is the total number of observations

p_i is the prediction

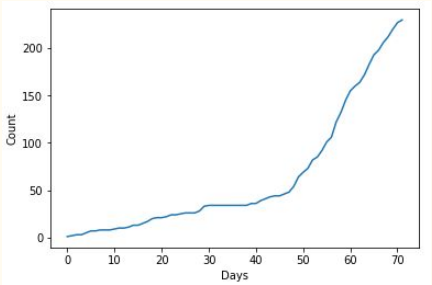
a_i is the actual value

log(x) is the natural logarithm of x

Methodology

1. Visualization
2. Pre-Processing
3. Linear Regression
4. Decision Tree
5. Random Forest
6. Gradient Boosted Tree
7. Higher Dimensional Data Projection + Gradient Boosted Tree
8. Post-Processing

Country_Region	count(ConfirmedCases)
China	830
US	505
Australia	290
Canada	248
France	193



	A	B	C
1	ForecastId	ConfirmedCases	Fatalities
2	1	273	6
3	2	281	6
4	3	299	7
5	4	349	7
6	5	367	11
7	6	423	14
8	7	444	14
9	8	484	15
10	9	521	15
11	10	555	18
12	11	607	18
13	12	665	21
14	13	714	23
15	14	714	23
16	15	714	23
17	16	714	23
18	17	714	23
19	18	714	23
20	19	714	23
21	20	714	23
22	21	714	23
23	22	714	23
24	23	714	23
25	24	714	23
26	25	714	23
27	26	714	23
28	27	714	23
29	28	714	23
30	29	714	23
31	30	714	23
32	31	714	23
33	32	714	23
34	33	714	23
35	34	714	23
36	35	714	23
37	36	714	23
38	37	714	23
39	38	714	23
40	39	714	23
41	40	714	23
42	41	714	23
43	42	714	23
44	43	714	23
45	44	277	16
46	45	304	17

In PySpark, it is not possible to train a regression model with multiple outputs, as a result of which, separate training and testing needs to be performed for Confirmed Cases and Fatalities followed by joining their individual outputs in a single file for submission

Visualization

Schemas

1. Province_State, Country_Region, Date, Confirmed Cases, Fatalities

2. Province_State, Country_Region, Date

```
root
 |-- Id: integer (nullable = true)
 |-- Province_State: string (nullable = true)
 |-- Country_Region: string (nullable = true)
 |-- Date: timestamp (nullable = true)
 |-- ConfirmedCases: double (nullable = true)
 |-- Fatalities: double (nullable = true)
```

```
root
 |-- ForecastId: integer (nullable = true)
 |-- Province_State: string (nullable = true)
 |-- Country_Region: string (nullable = true)
 |-- Date: timestamp (nullable = true)
```

Countries and Days

1. 180/72
2. 180/43

Countries and Days

Train

```
train_df.createOrReplaceTempView("train_df")
sqlDF = spark.sql("SELECT Country_Region FROM train_df GROUP BY Country_Region")
print ("Total Countries: ",len(sqlDF.toPandas()["Country_Region"]))
sqlDF = spark.sql("SELECT Date FROM train_df GROUP BY Date")
print ("Total Days: ",len(sqlDF.toPandas()["Date"]))
```

```
Total Countries:  180
Total Days:  72
```

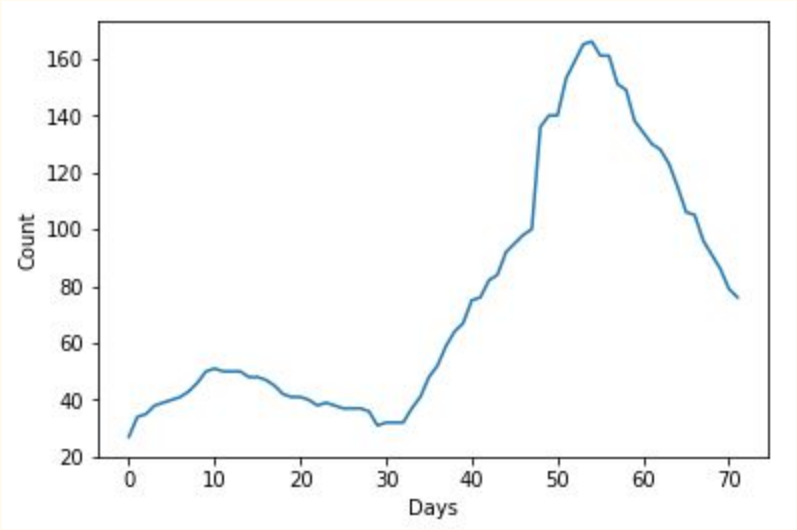
Test

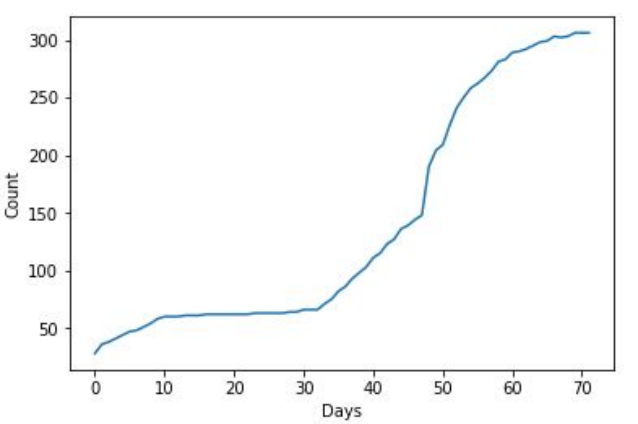
```
test_df.createOrReplaceTempView("test_df")
sqlDF = spark.sql("SELECT Country_Region FROM test_df GROUP BY Country_Region")
print ("Total Countries: ",len(sqlDF.toPandas()["Country_Region"]))
sqlDF = spark.sql("SELECT Date FROM test_df GROUP BY Date")
print ("Total Days: ",len(sqlDF.toPandas()["Date"]))
```

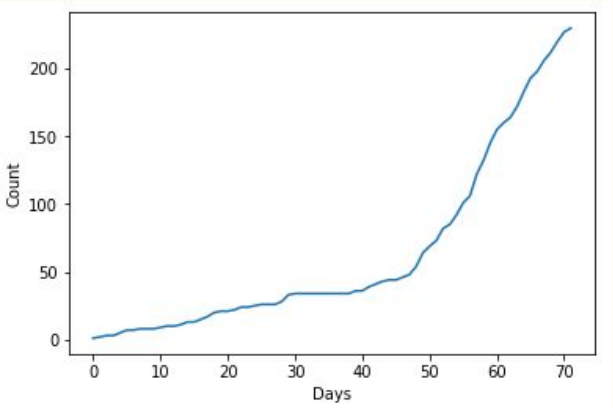
```
Total Countries:  180
Total Days:  43
```

Confirmed Cases and Fatalities (Days)

1. Confirmed Cases but no Fatalities
2. Confirmed Cases
3. Fatalities







Confirmed Cases and Fatalities (Countries)

1. Confirmed Cases but no Fatalities
2. Confirmed Cases
3. Fatalities

Country_Region	count(ConfirmedCases)
China	830
US	505
Australia	290
Canada	248
France	193

Country_Region	count(ConfirmedCases)
Panama	1
Botswana	1
Albania	2
Guatemala	2
Sierra Leone	3

Country_Region	count(Fatalities)
China	1527
US	740
Canada	86
Australia	86
France	85

Country_Region	count(ConfirmedCases)
China	2357
US	1245
Australia	376
Canada	334
France	278

Country_Region	count(ConfirmedCases)
Sierra Leone	3
Burundi	3
Botswana	4
MS Zaandam	6
Burma	7

Country_Region	count(Fatalities)
Libya	1
Congo (Brazzaville)	1
Zambia	1
Senegal	2
MS Zaandam	2

Province_State had NULL values

1. Remove records having NULL values
2. Remove Province_State as a feature

Linear Regression	Including Province State	Without Including Province State
Score	3.65539	3.35323

TABLE I
RESULTS USING LINEAR REGRESSION

Pre-Processing

1. Remove NULL values
2. Convert Timestamp to Unix Timestamp
3. Convert Categorical Attributes to Nominal

Remove NULL values

```
N_NULL_DF = spark.sql("SELECT * FROM train_df WHERE Province_State IS
NOT NULL")
```

Convert Timestamp to UnixTimestamp

```
indexer = StringIndexer(inputCol="Province_State",
outputCol="Province_StateIndex")
indexed = indexer.fit(N_NULL_DF).transform(N_NULL_DF)
indexed.show()
```

Convert Categorical Attributes to Nominal

```
indexer = StringIndexer(inputCol="Province_State",
outputCol="Province_StateIndex")
indexed = indexer.fit(N_NULL_DF).transform(N_NULL_DF)
indexed.show()

indexer = StringIndexer(inputCol="Country_Region",
outputCol="Country_RegionIndex")
indexed = indexer.fit(indexed).transform(indexed)
indexed.show()
```

Linear Regression

Linear regression is a basic and commonly used type of predictive analysis. The overall idea of regression is to examine two things: (1) does a set of predictor variables do a good job in predicting an outcome (dependent) variable? (2) Which variables in particular are significant predictors of the outcome variable, and in what way do they indicated by the magnitude and sign of the beta estimates–impact the outcome variable?

These regression estimates are used to explain the relationship between one dependent variable and one or more independent variables. The simplest form of the regression equation with one dependent and one independent variable is defined by the formula y = c + b*x, where y = estimated dependent variable.

Linear Regression	Including Province State	Without Including Province State
Score	3.65539	3.35323

TABLE I

RESULTS USING LINEAR REGRESSION

Decision Tree

A tree can be “learned” by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called recursive partitioning. The recursion is completed when the subset at a node all has the same value of the target variable, or when splitting no longer adds value to the predictions. The construction of decision tree classifier does not require any domain knowledge or parameter setting, and therefore is appropriate for exploratory knowledge discovery. Decision trees can handle high dimensional data. In general decision tree classifier has good accuracy. Decision tree induction is a typical inductive approach to learn knowledge on classification.

Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance. An instance is classified by starting at the root node of the tree,testing the attribute specified by this node,then moving down the tree branch corresponding to the value of the attribute. This process is then repeated for the subtree rooted at the new node.

Decision Tree	Depth = 3	Depth = 5
Score	2.49157	2.38298

TABLE II

RESULTS USING DECISION TREE

Random Forest

Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction.

Random Forest	Trees = 2	Trees = 20	Trees = 100
Score	3.13457	3.15120	3.20820

TABLE III
RESULTS USING RANDOM FOREST

Gradient Boosted Tree

A gradient boosted model is an ensemble of either regression or classification tree models. Both are forward-learning ensemble methods that obtain predictive results through gradually improved estimations. Boosting is a flexible nonlinear regression procedure that helps improving the accuracy of trees. By sequentially applying weak classification algorithms to the incrementally changed data, a series of decision trees are created that produce an ensemble of weak prediction models. While boosting trees increases their accuracy, it also decreases speed and human interpretability. The gradient boosting method generalizes tree boosting to minimize these issues.

Depth	Score
3	2.54647
5	2.05806
7	2.01467
9	1.98333
30	1.98171

TABLE IV
RESULTS USING GRADIENT BOOSTED TREE

Higher Dimensional Data Projection + Gradient Boosted Tree

In this experiment, I tried projecting data to higher dimensions by first taking square and then cube of Country_RegionIndex followed by Gradient Boosted Tree.

HDDP + GBT	Square	Cube
Score	2.2225	2.22225

TABLE V

RESULTS USING HIGHER DIMENSION DATA PROJECTION + GRADIENT BOOSTED TREE

Post-Processing

Post-processing step included the creation of submission file by combining the outputs of multiple files including a separate file each for confirmed cases and fatalities

Conclusion

I found out that Gradient Boosted Tree performed the best among the algorithms tested with a score of 1.98171 on week-3 and 2.22225 on week-4 which was the top score on the private leaderboard as the competition was closed.

Future Work

It will be a good direction to try different values for records having NULL values and see the effect.

References

https://www.statisticssolutions.com/what-is-linear-regression/

https://www.geeksforgeeks.org/decision-tree/

https://towardsdatascience.com/understanding-random-forest-58381e0602d2

https://docs.rapidminer.com/latest/studio/operators/modeling/predictive/trees/gradient_boosted_trees.html