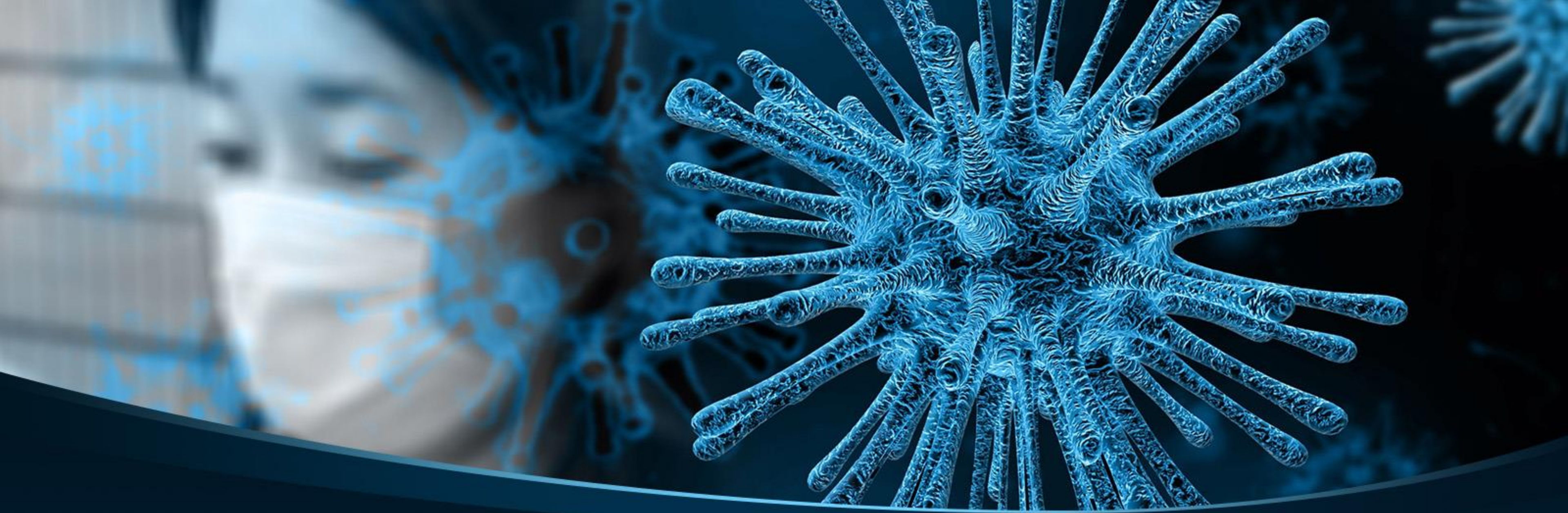


Predicting the Spread of COVID19

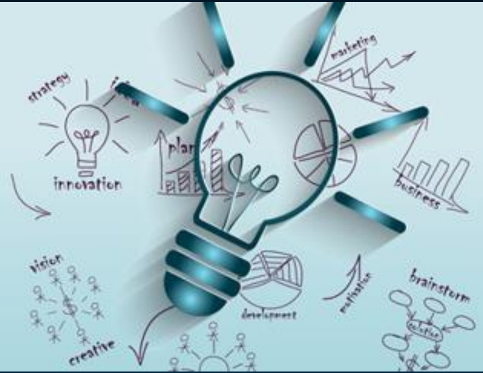
DSE 6300: Data Science Application Development

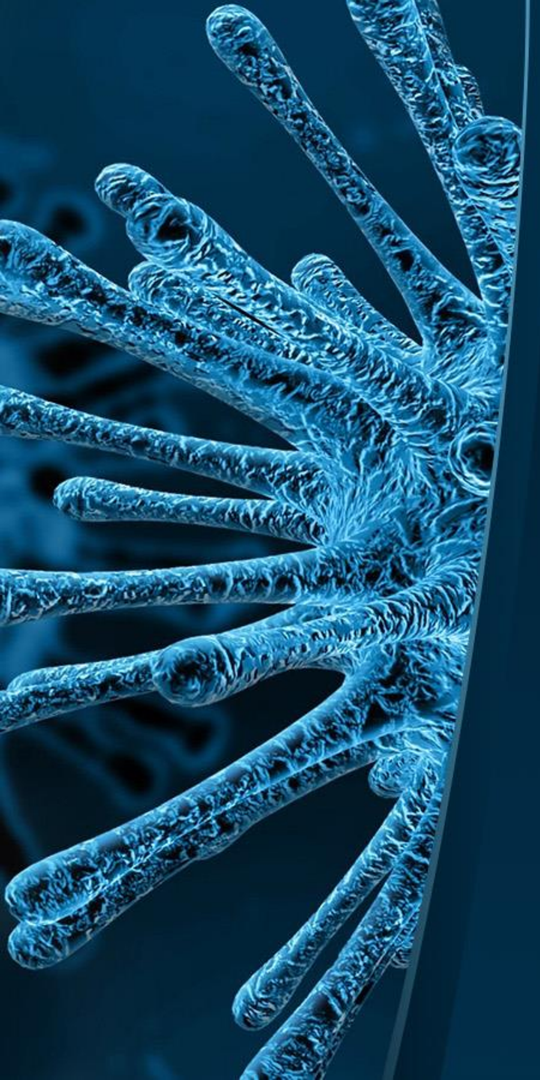
By: Kyle W. Brown, Ishaan Khurana, Jacqueline Connolly, Nishchitha Manjunath, Derick Karolak

4/24/2020

Overview

- Background
- Data
- Pipeline
- Models
- Visualization
- Summary





Data collection/cleaning

- John Hopkins daily case/death data for each county in US and each country internationally
- Collected predictor data by county (unemployment rate, poverty rate, mortality rates) joined with county case data for regression models

Pipeline

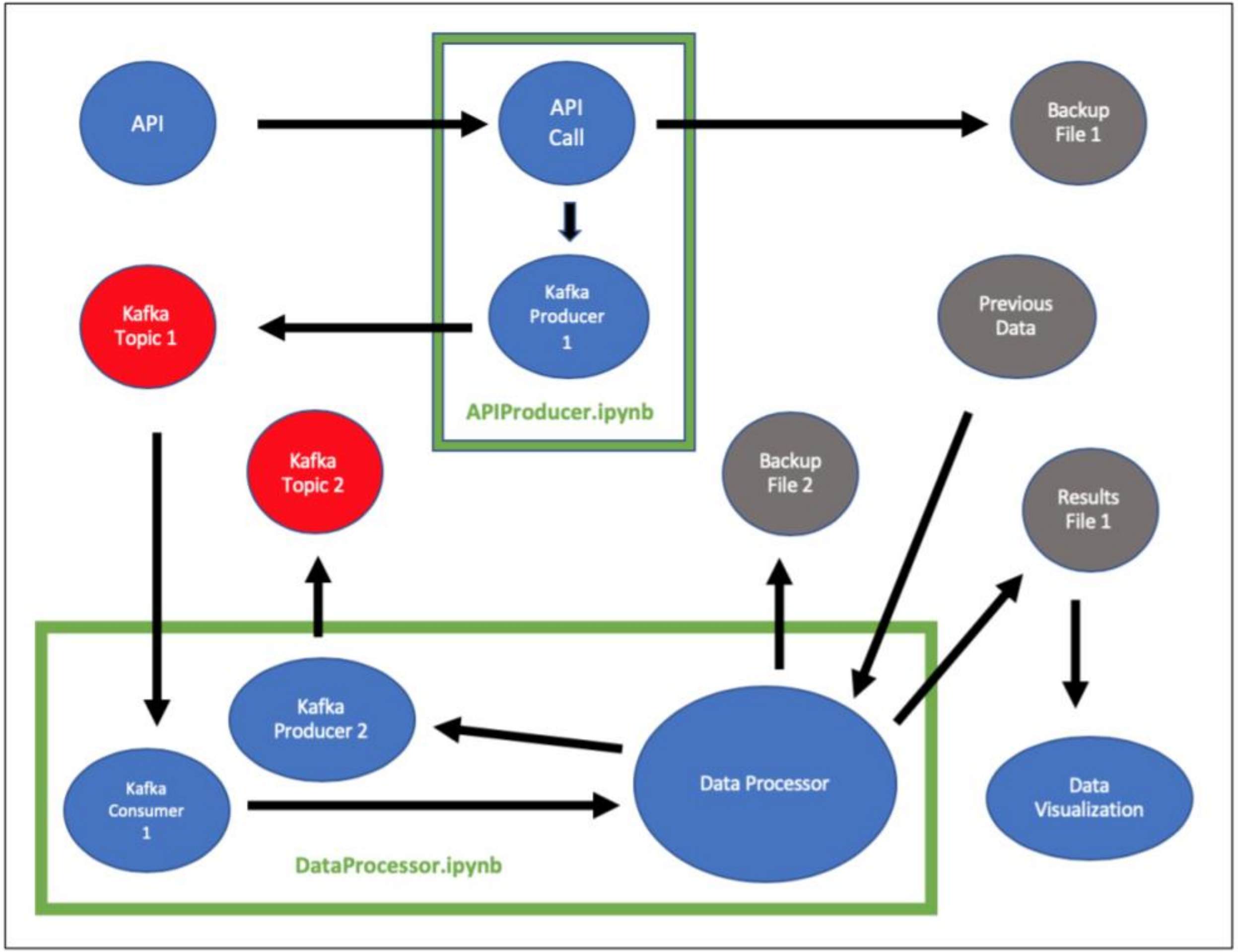
Disclaimer:

Would have done much more with more time!

Models/Visualizations

Utilizing the Grid

Integrating Spark



Setup

- Navigate to Kafka Home Directory
- Run Zookeeper
  - bin/zookeeper-server-start.sh config/zookeeper.properties
- Run Kafka Broker
  - bin/kafka-server-start.sh config/server.properties
- Open Python Scripts
- Run Notebooks
- Navigate to Project Directory
- Start Local Server
  - http-server &
- Open Server Port in Browser
  - http://localhost:8082
- Open HTML Page with Visualizations

First Python Script: API Call

Make API Call

Write Data to Backup File

Define Kafka Producer

Write Data to Kafka Topic: covid-rawData

Runs Once an Hour

```
#Import Libraries
import requests
import pandas as pd
from time import sleep
from json import dumps
from kafka import KafkaProducer

#Define Request Parameters for API
url = "https://covid-193.p.rapidapi.com/statistics"
headers = {
    "x-rapidapi-host": "covid-193.p.rapidapi.com",
    "x-rapidapi-key": "00d9966b6fmshe53ec07961c056ep117aeejsn5ca90b187893"
}

#Define Kafka Producer
producer = KafkaProducer(bootstrap_servers=['localhost:9092'],
                         value_serializer=lambda x:
                         dumps(x).encode('utf-8'))
```

```
while(True):
    #Make API Call
    r = requests.get(url, headers = headers)

    #Parse JSON Response
    data = r.json()

    #Convert to Dataframe and Write to CSV
    data2 = pd.DataFrame(data['response'])
    data2.to_csv('fromAPI.csv')

    #Write data to Kafka Topic
    producer.send('covid-rawData', value=data)

    #Allow program to sleep until next check
    sleep(60*60)
```

DataProcessor.ipynb

Consumes Data From Kafka Topic

Isolates New Cases Data
Writes to CSV

Isolates Total Cases Data
Writes to CSV

Import Old Data

Clean Country Names

Combine New and Old Data

Write to File

Write to Kafka Topic

```
#Import Libraries
from kafka import KafkaConsumer, KafkaProducer
from json import loads
import pandas as pd
import numpy as np
import json
from json import dumps
from time import sleep

pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)

#Define Kafka Consumer
consumer = KafkaConsumer(
    'covid-rawData',
    bootstrap_servers=['localhost:9092'],
    api_version=(0,10),
    consumer_timeout_ms = 5000,
    auto_offset_reset = 'earliest'
)

#Define Kafka Producer
producer = KafkaProducer(bootstrap_servers=['localhost:9092'],
                         value_serializer=lambda x:
                         x.encode('utf-8'))
```

DataProcessor.ipynb

```
while(True):
    #Run Consumer and Record Data
    output = []
    for message in consumer:
        message = message.value
        output.append(message)

    #Parse the JSON data
    parsed_json = (json.loads(output[0]))

    #Create DataFrame
    df = pd.DataFrame(parsed_json['response'])

    #Isolate and Format New Cases Data
    new = []
    for i in range(len(parsed_json['response'])):
        n = parsed_json['response'][i]['cases']['new']
        if n == None:
            n = '+0'
        new.append(int(n.replace('+', '')))

    #Create DataFrame with New Cases
    df2 = pd.concat([df['country'], pd.DataFrame(new)], axis = 1)
    df2.columns = ['country', 'new']

    #Isolate Top 10 New Cases and Write to CSV
    top10New = df2.sort_values(by='new', ascending=False).iloc[:10, :]
    top10New.to_csv('top10New.csv')

    #Isolate Totals Data
    totals = []
    for i in range(len(parsed_json['response'])):
        totals.append(int(parsed_json['response'][i]['cases']['total']))

    #Create DataFrame with Totals Data
    df3 = pd.concat([df['country'], pd.DataFrame(totals)], axis = 1)
    df3.columns = ['country', 'total']

    #Identify Top 10 and Write to CSV
    top10Total = df3.sort_values(by = 'total', ascending = False).iloc[3:, :].head(12)[(df3['country'] != 'Asia') & (df
    top10Total.to_csv('top10Totals.csv')

    #Read in start data
    startData = pd.read_csv('GlobalConfirmed.txt')

    #Subselect data needed
    startData2 = pd.concat([startData['Country/Region'], startData['1/22/20']], axis = 1)
    startData2.columns = ['country', 'day1']
```

```
    #Group by country
    startData3 = pd.DataFrame(startData.groupby('Country/Region').sum()['1/22/20'])

    #Clean country names
    cs = []
    for i in range(len(df3)):
        c = df3['country'][i].replace("-", " ")
        cs.append(c)
    df3['country2'] = cs
    df3['country3'] = df3['country2'].replace("USA", "US").replace("S Korea", "Korea, South").replace('Guinea Bissau',

    #Combine new and old data
    df4 = pd.merge(startData3.reset_index(), df3, left_on = ['Country/Region'], right_on = ['country3'])

    df5 = pd.concat([df4['Country/Region'], df4['1/22/20'], df4['total']], axis = 1)
    df5.columns = ['country', 'day1', 'day2']

    #Subselect top 10
    df6 = df5.sort_values(by = 'day2', ascending = False).head(10)

    #Write to file
    df6.to_csv("twoDays.csv")

    #Convert DataFrame to JSON to be sent to Kafka
    dataJSON = df6.reset_index(drop=True).to_json(orient='records')

    #Write data to Kafka Topic
    producer.send('covid-cleanData', value=dataJSON)

    #Sleep for an hour
    sleep(60*60)
```

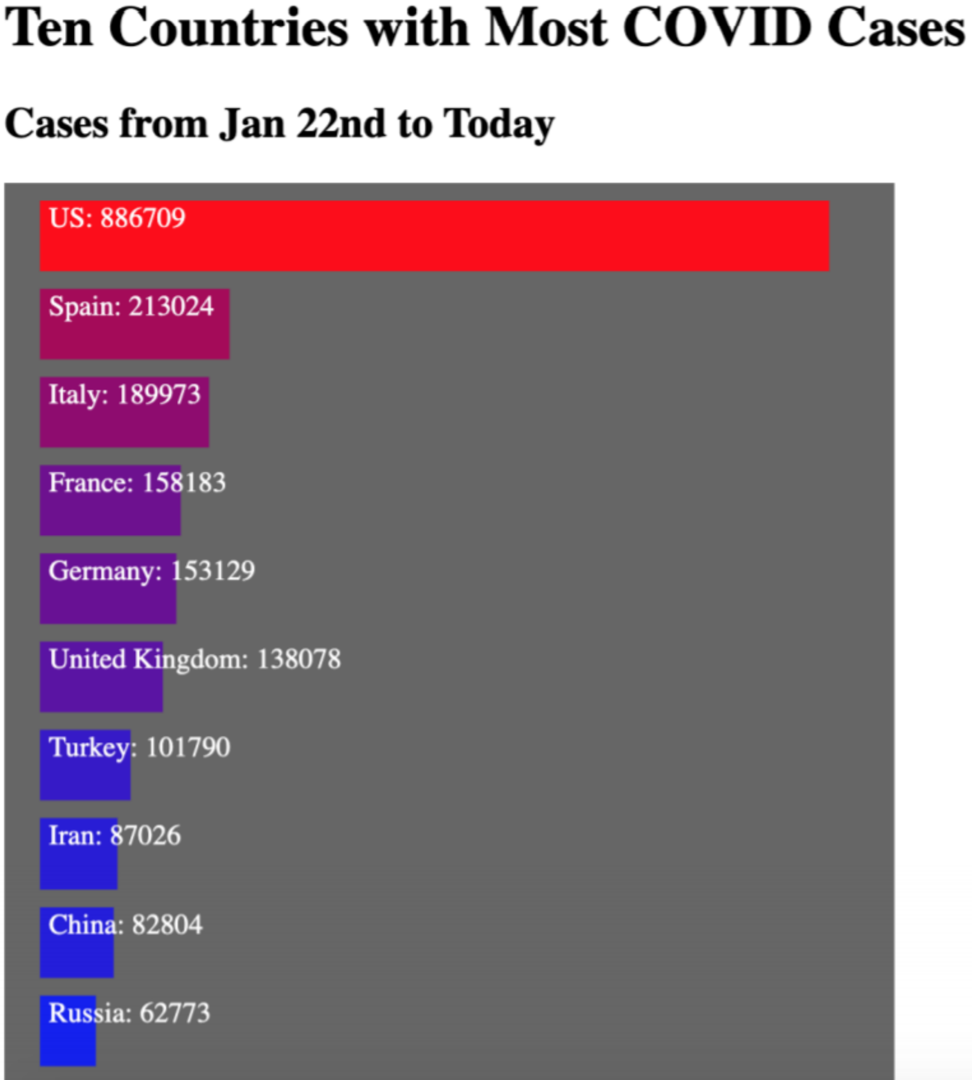
Data Visualizations: D3

```
<!DOCTYPE html>
<html>
<head>
  <title>D3</title>
  <script src="http://d3js.org/d3.v3.min.js"></script>
</head>
<body>
  <h1>Ten Countries with Most COVID Cases</h1>
<h2></h2>
  <script>

    data = [100, 100, 100, 100, 100, 100, 100, 100, 100, 100]

    var widthScale = d3.scale.linear()
      .domain([0, 1000000])
      .range([0, 500]);

    var color = d3.scale.linear()
      .domain([50000, 300000])
      .range(['blue', 'red']);

    var canvas = d3.select("body")
      .append("svg")
      .attr("width", 500)
      .attr("height", 520)
      .style("background-color", "#666666");

    var subHeading = d3.select("h2").text("Cases from Jan 22nd to Today");

    var bars = canvas.selectAll("rect")
          .data(data)
          .enter()
            .append("rect")
            .attr("width", function(d) { return widthScale(d); })
            .attr("height", 40)
            .attr("y", function (d, i) { return i * 50 + 10; })
            .attr("x", 20)
            .attr("fill", "yellow");


    var countries = canvas.selectAll("text")
          .data(data)
          .enter()
            .append("text")
            .attr("fill", "white")
            .attr("y", function (d, i) { return i * 50 + 25; })
            .attr("x", 25)
            .text(function (d) { return ""; });
```

```
function updateData() {

  d3.csv("twoDays.csv", function(data2) {

    setInterval(function() {

      d3.csv("twoDays.csv", function(data3) {

        var bars3 = canvas.selectAll("rect")
      .data(data3)
      .enter()
        .append("rect")
        .attr("width", function(d) { return widthScale(d.day1); })
        .attr("height", 40)
        .attr("y", function (d, i) { return i * 50 + 10; })
        .attr("x", 20)
        .attr("fill", "blue");

      var countries3 = canvas.selectAll("text")
      .data(data3)
      .enter()
        .append("text")
        .attr("fill", "white")
        .attr("y", function (d, i) { return i * 50 + 25; })
        .attr("x", 25)
        .text(function (d) { return d.country + ": " + d.day1; });

        bars.transition()
          .attr("width", function(d) { return widthScale(d.day1); })
          .duration(2000)
          .attr("fill", function(d) { return color(d.day1); })
          .transition()
            .duration(2000)
            .attr("width", function(d) { return widthScale(d.day2); })
            .attr("fill", function(d) { return color(d.day2); });

        countries.transition()
          .text(function (d) { return d.country + ": " + d.day1; })
          .duration(2000)
          .transition()
            .text(function (d) { return d.country + ": " + d.day2; });

      });

    }, 8000);
```

Data Visualizations: D3

```
        var bars2 = canvas.selectAll("rect")
          .data(data2)
          .enter()
            .append("rect")
            .attr("width", function(d) { return widthScale(d.day1); })
            .attr("height", 40)
            .attr("y", function (d, i) { return i * 50 + 10; })
            .attr("x", 20)
            .attr("fill", "blue");

        var countries2 = canvas.selectAll("text")
          .data(data2)
          .enter()
            .append("text")
            .attr("fill", "white")
            .attr("y", function (d, i) { return i * 50 + 25; })
            .attr("x", 25)
            .text(function (d) { return d.country + ": " + d.day1; });

        bars2.transition()
          .attr("width", function(d) { return widthScale(d.day2); })
          .duration(2000)
          .attr("fill", function(d) { return color(d.day2); });

        countries2.transition()
          .text(function (d) { return d.country + ": " + d.day2; })
          .duration(2000);

      });

    }

    updateData();

  </script>
</body>
</html>
```



Demo

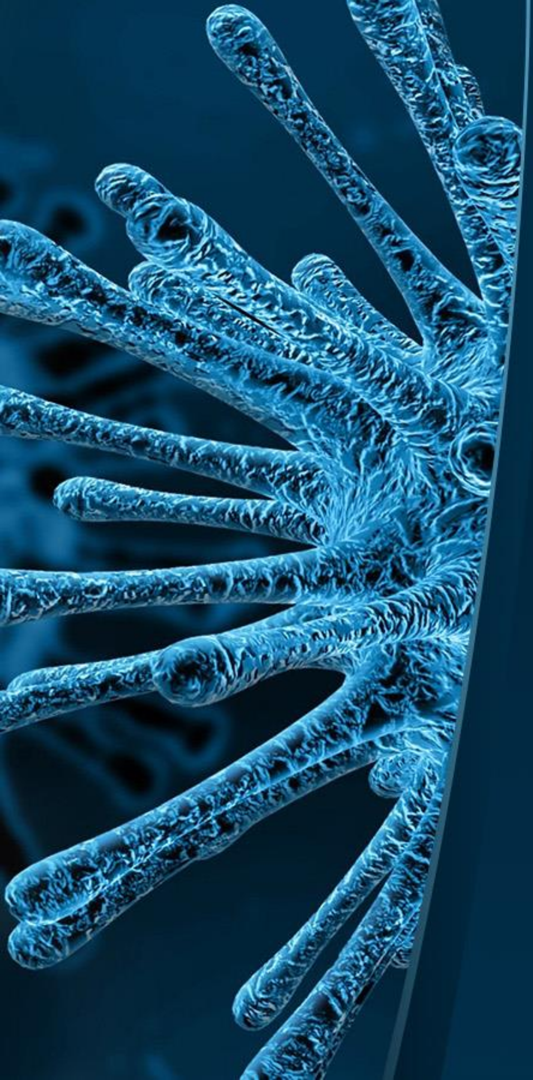
Model Summaries

Python Time series models by each county, country

```
bestMSE=10000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
bestP1=0
bestP2=0
bestP3=0
for p1 in range(6):
    for p2 in range(6):
        for p3 in range(6):
            dict_of_sources = dict(iter(UScases.groupby('Province_State')))
            sumMSE= 0
            for x, y in dict_of_sources.items():
                ActualData= y['Value'].values.astype(float)
                singular= sum(ActualData)
                if(singular>2) :
                    NumberOfElements = len(ActualData)
                    TrainingSize = int(NumberOfElements * 0.7)
                    TrainingData = ActualData[0:TrainingSize]
                    TestData = ActualData[TrainingSize:NumberOfElements]
                    #model = ARIMA(TestData, order=(1, 1, 0))
                    model = SARIMAX(ActualData, trend='c', order=(p1,p2,p3), enforce_stationarity=False, enforce_invertibility=False)
                    model_fit = model.fit()
                    prediction = model_fit.forecast(len(TestData))
                    #plt.figure(figsize=(10,5))
                    #plt.plot(TestData,prediction, color='red')
                    MSEArima= (sum(prediction[0]-TestData)**2)/len(TestData)
                    sumMSE= sumMSE+MSEArima
                    #print(prediction)
                    print(MSEArima,x)
            print("SUM of MSE",sumMSE,"P1 value:",p1,"P2 value",p2,"P3 value",p3)
            if(sumMSE<bestMSE):
                sumMSE= bestMSE
                bestP1=p1
                bestP2=p2
                bestP3=p3
```

```
UScases= pd.read_excel('CoronaUSCasesTransposed.xlsx')
USDeaths= pd.read_excel('CoronaUSDeathsTransposed.xlsx')
pd.set_option('display.max_columns', None)
UScases.head(10)
```

	UID	iso2	iso3	code3	FIPS	Admin2	Province_State	Country_Region	Lat	Long_	Combined_Key	Name	Value
0	16	AS	ASM	16	60.0	NaN	American Samoa	US	-14.271	-170.132	American Samoa, US	1/22/20	0
1	16	AS	ASM	16	60.0	NaN	American Samoa	US	-14.271	-170.132	American Samoa, US	1/23/20	0
2	16	AS	ASM	16	60.0	NaN	American Samoa	US	-14.271	-170.132	American Samoa, US	1/24/20	0
3	16	AS	ASM	16	60.0	NaN	American Samoa	US	-14.271	-170.132	American Samoa, US	1/25/20	0
4	16	AS	ASM	16	60.0	NaN	American Samoa	US	-14.271	-170.132	American Samoa, US	1/26/20	0
5	16	AS	ASM	16	60.0	NaN	American Samoa	US	-14.271	-170.132	American Samoa, US	1/27/20	0
6	16	AS	ASM	16	60.0	NaN	American Samoa	US	-14.271	-170.132	American Samoa, US	1/28/20	0
7	16	AS	ASM	16	60.0	NaN	American Samoa	US	-14.271	-170.132	American Samoa, US	1/29/20	0
8	16	AS	ASM	16	60.0	NaN	American Samoa	US	-14.271	-170.132	American Samoa, US	1/30/20	0
9	16	AS	ASM	16	60.0	NaN	American Samoa	US	-14.271	-170.132	American Samoa, US	1/31/20	0

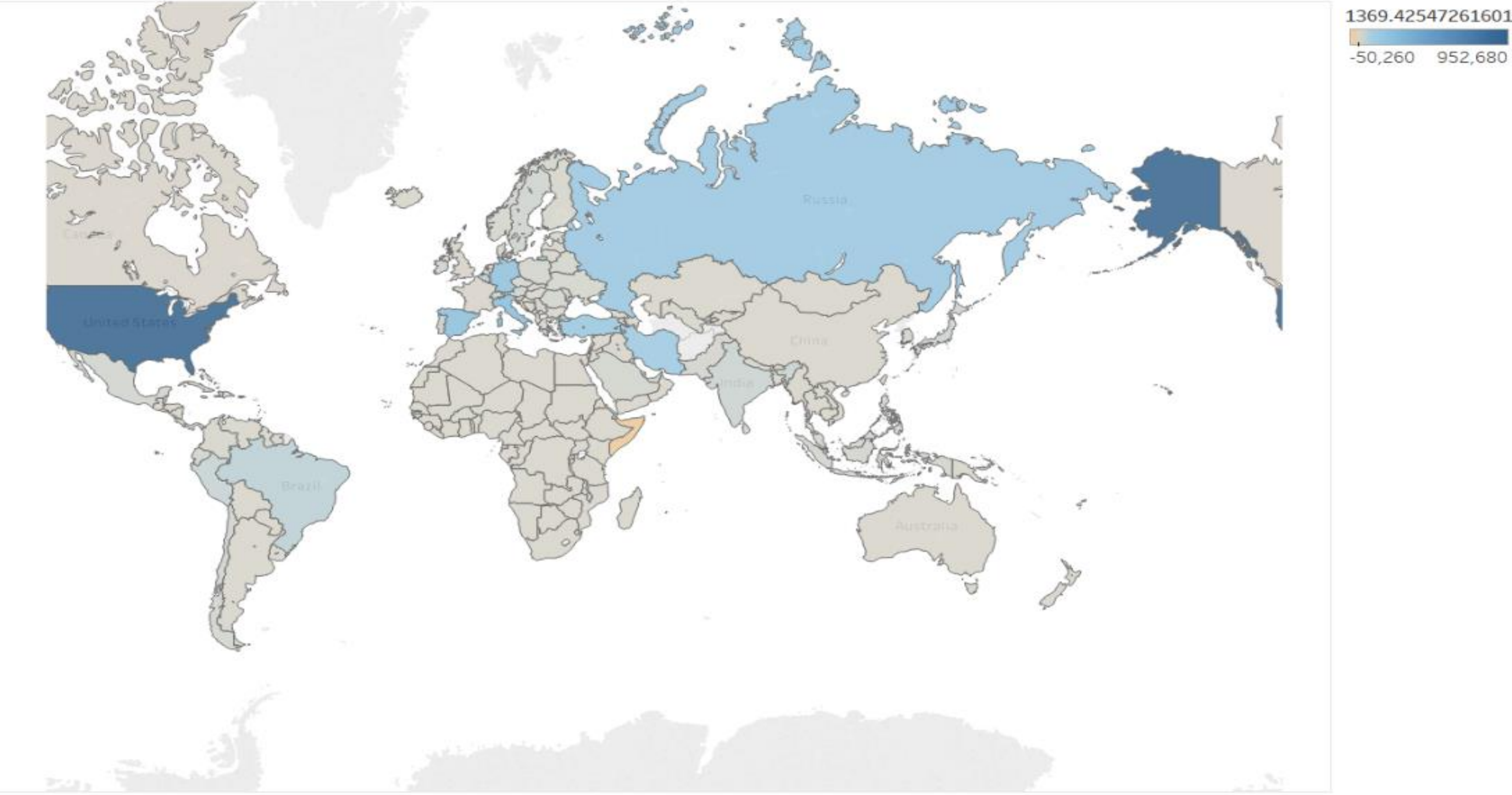


Model Results

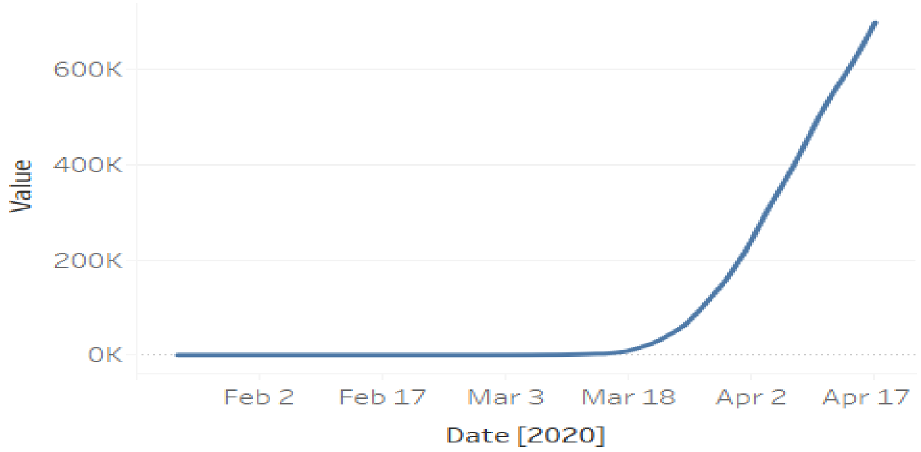
- Time series shows a steady increase everywhere through May 1 – Not able to capture when the peak hits for any county in the US or any country
- Predictions do not appear accurate for every country, county – unable to select optimal model parameters for each individual dataset (data split by location)

<April 13 Cases by Country>

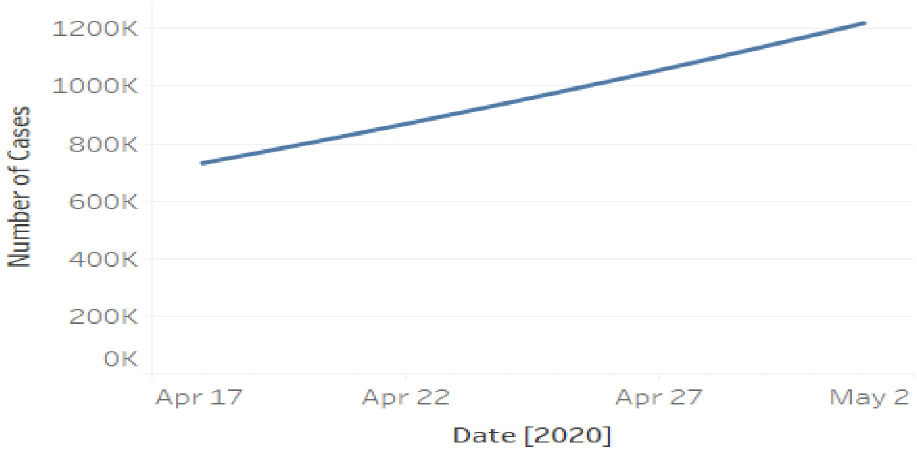
<May 1 Predictions by Country>



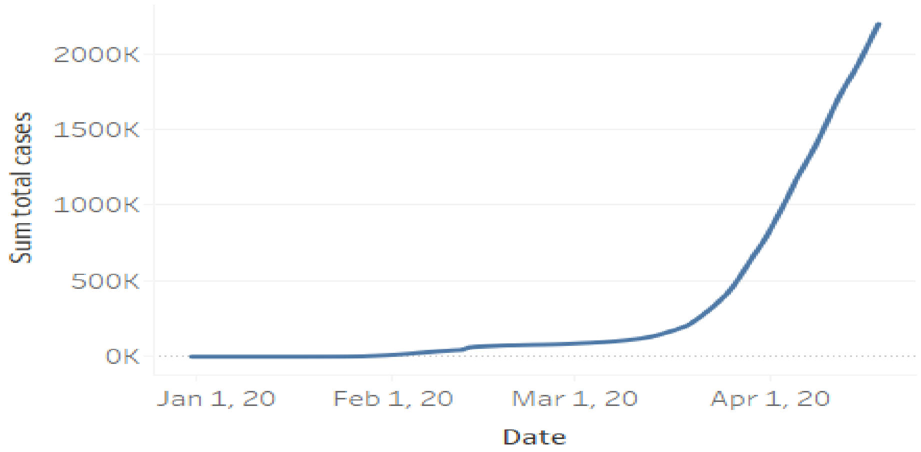
<US Total Cases >



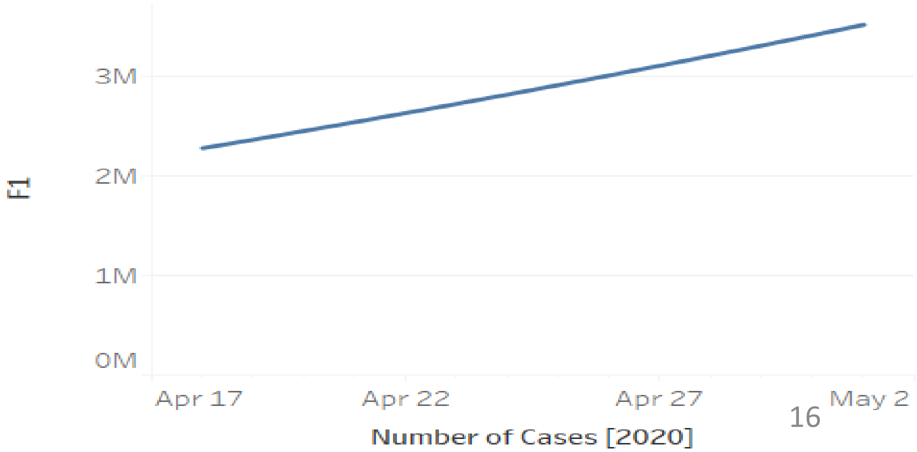
<US Total Case Predictions>

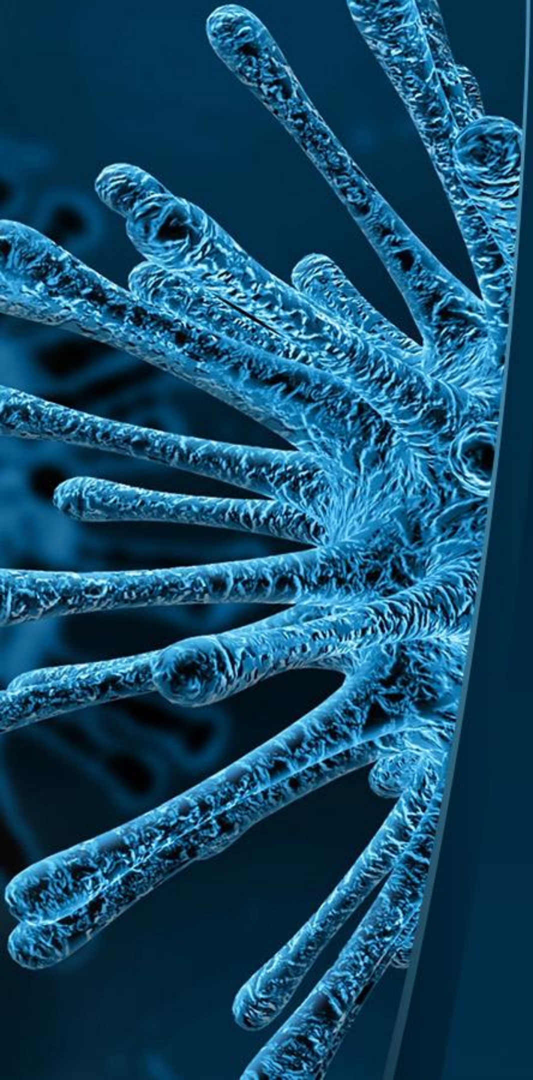


<World Total Cases>



<World Total Case Prediction>





Models

- Regressions
  - Linear
  - Mulivariate
  - Non-Linear
- Principal Component Analysis
- Time Series
- AutoML Deep Learning

Multivariable Regression

```
# Linear Regression of JHU Time Series

# Mortal Rate as the Predictor
mortalUSlm <- read.csv("https://raw.githubusercontent.com/WorldCapital/COVID19-Project/master/COVID-19/JHU_county
mortalUSlm <- lm(Mortality_Rate ~ Confirmed + Deaths + Recovered + FIPS + Incident_Rate  + People_Tested  + Peop
confint(mortalUSlm)
summary(mortalUSlm)
#Multiple R-squared:  0.6843, Adjusted R-squared:  0.5264
#F-statistic: 4.334 on 9 and 18 DF,  p-value: 0.003951


mse = mean(mortalUSlm$residuals^2)
print(paste0("MSE= ", mse))

#[1] "MSE= 0.717658874682637"

print(paste0("RMSE= ", RMSE(mortalUSlm$residuals)))

#[1] "RMSE= 0.847147492873961"
```

Non-Linear Regressions

```
#Non-Linear Model NY Times using Deaths
nyTimesNLM <- read.csv("https://raw.githubusercontent.com/WorldCapital/COVID19-Project/master/COVID-19/nyt-us-co
nyTimesNLM <- lm(deaths ~ fips + cases, I(cases^2), data = ny_Times_Counties)
confint(nyTimesNLM)
summary(nyTimesNLM)
#Residual standard error: 0.6038 on 55656 degrees of freedom
#(3543 observations deleted due to missingness)
#Multiple R-squared:  0.9265, Adjusted R-squared:  0.9265
#F-statistic: 3.51e+05 on 2 and 55656 DF,  p-value: < 2.2e-16

mse = mean(nyTimesNLM$residuals^2)
print(paste0("MSE= ", mse))

#[1] "MSE= 0.364578917674488"

print(paste0("RMSE= ", RMSE(nyTimesNLM$residuals)))

#[1] "RMSE= 0.603803707900579"
```

AutoML Models

```
#Split data into Train/Validation/Test Sets
split_h2o <- h2o.splitFrame(conv_data.hex, c(0.6, 0.2), seed = 1234 )
train_conv_h2o <- h2o.assign(split_h2o[[1]], "train" ) # 60%
valid_conv_h2o <- h2o.assign(split_h2o[[2]], "valid" ) # 20%
test_conv_h2o  <- h2o.assign(split_h2o[[3]], "test" )  # 20%


#Model
# Set names for h2o
target <- "cases_cum"
predictors <- setdiff(names(train_conv_h2o), target)
# Run the automated machine learning
automl_h2o_models <- h2o.automl(
  x = predictors,
  y = target,
  training_frame    = train_conv_h2o,
  leaderboard_frame = valid_conv_h2o
)
```

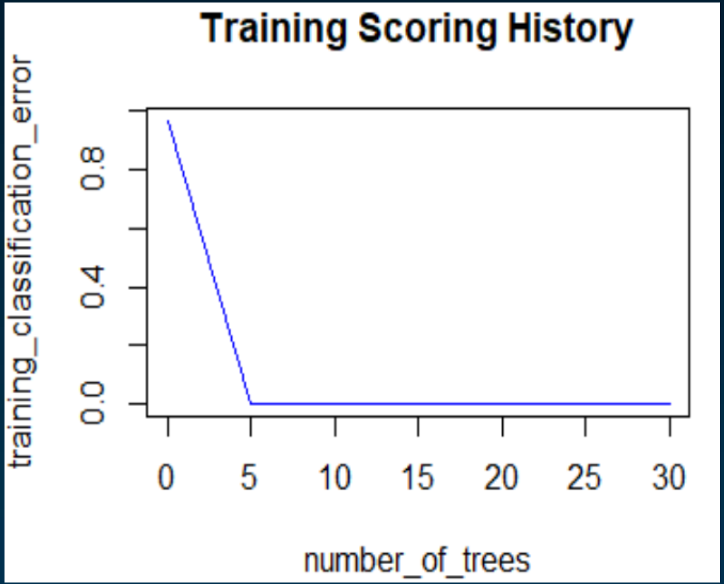
```
#Cross-Validation Metrics Summary:
#                             mean          sd cv_1_valid cv_2_valid cv_3_valid cv_4_valid cv_5_valid
#accuracy               0.30769232 0.094211146 0.46153846 0.23076923 0.23076923 0.30769232 0.30769232
#err                     0.6923077 0.094211146 0.53846157  0.7692308  0.7692308  0.6923077  0.6923077
#err_count                     9.0   1.2247449        7.0       10.0       10.0        9.0        9.0
#logloss                 3.4635184   0.4602315  2.7115996  3.6106849  3.9662728   3.548047  3.4809875
#max_per_class_error           1.0         0.0        1.0        1.0        1.0        1.0        1.0
#mean_per_class_accuracy 0.86153847  0.01884223  0.8923077 0.84615386 0.84615386 0.86153847 0.86153847
#mean_per_class_error    0.13846155  0.01884223 0.10769231 0.15384616 0.15384616 0.13846155 0.13846155
#mse                     0.6727001  0.08535113 0.53059614 0.71209335  0.7577791  0.6815926 0.68143946
#r2                      0.9982088 6.2457175E-4  0.9988421 0.99843067 0.99719256 0.99811006  0.9984687
#rmse                   0.81877226 0.053760055 0.72842026  0.8438563  0.8705051  0.8255862 0.82549345
```

Model Results

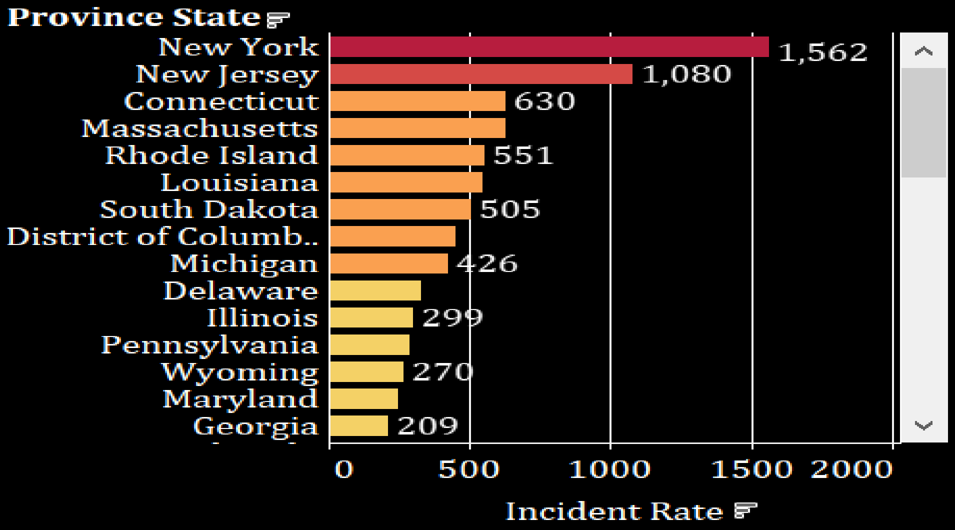
Model	r^2	rmse	mse
DeepLearning_grid__2_AutoML_20200419_000459_model_6	99.80%	87.24%	76.10%
GBM_grid__1_AutoML_20200419_000459_model_3	99.80%	84.66%	71.67%
GBM_grid__1_AutoML_20200419_000459_model_19	99.80%	80.01%	64.02%
GBM_grid__1_AutoML_20200419_000459_model_11	99.80%	75.51%	57.02%
GBM_grid__1_AutoML_20200419_000459_model_14	99.80%	75.48%	56.97%
GBM_grid__1_AutoML_20200419_000459_model_17	99.80%	74.26%	55.15%
Mortality_Rate_JHU_lm	52.64%	71.77%	84.72%
cases_cum_gbm_AutoML	99.81%	67.69%	82.16%
deaths_cum_gbm_AutoML	99.76%	49.60%	70.25%
Deaths_JHU_nlm	92.65%	36.46%	60.38%

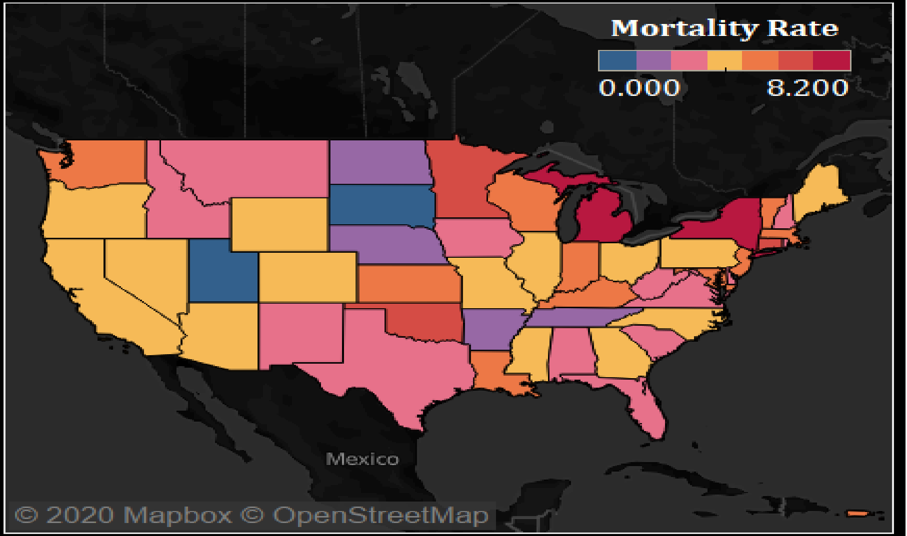
GBM Multinomial AutoML Leader

cases_cum_gbm_model

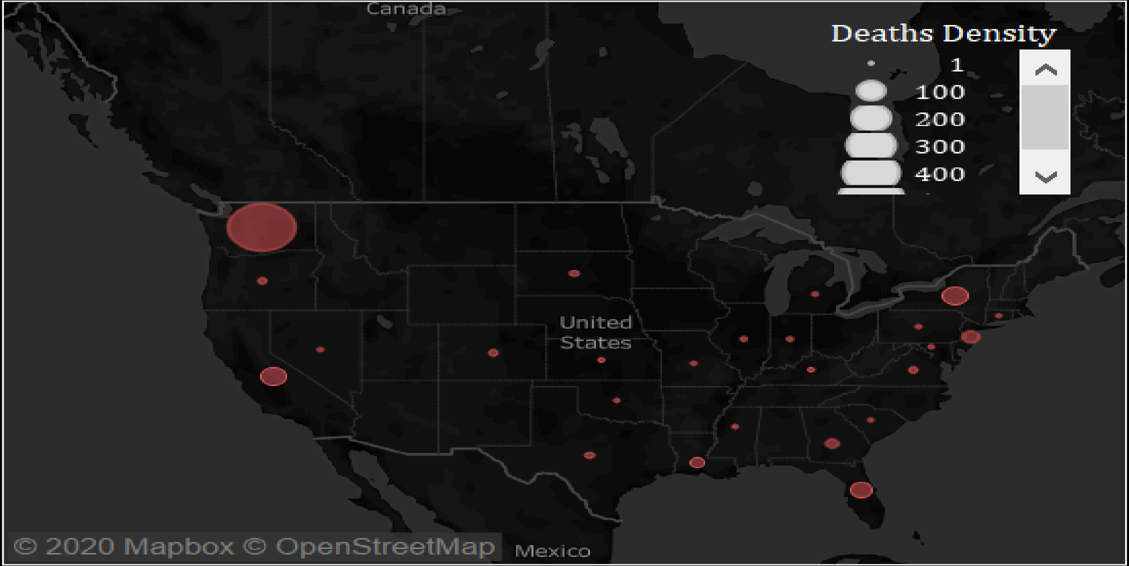


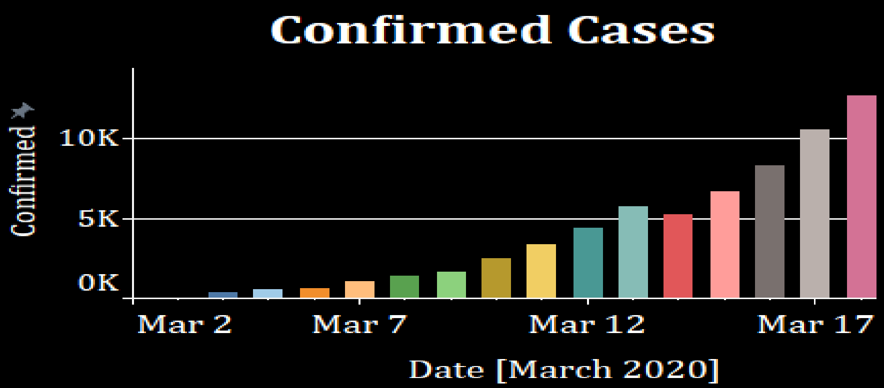
COVID-19 U.S.A. Dashboard

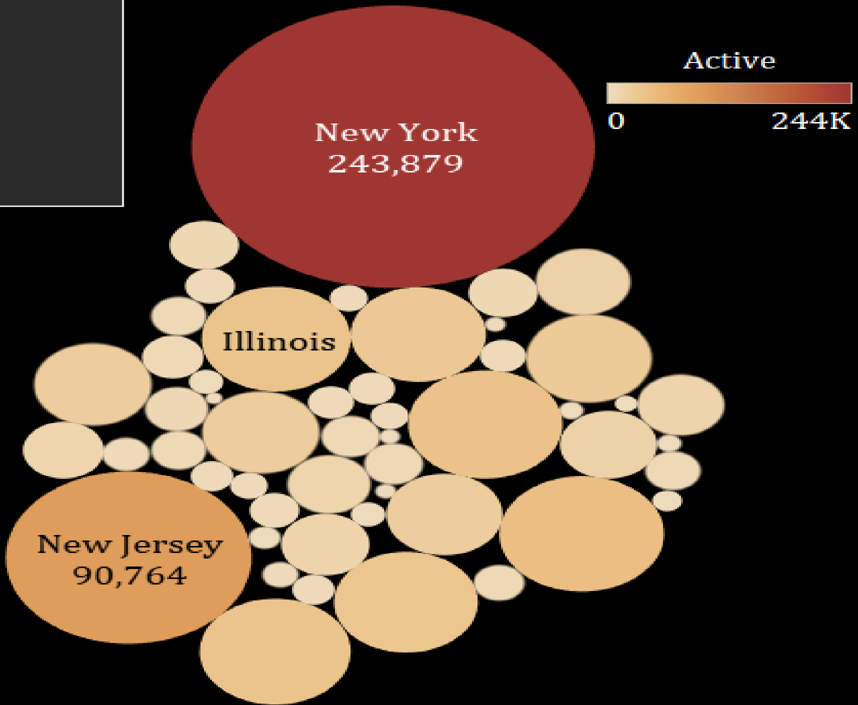


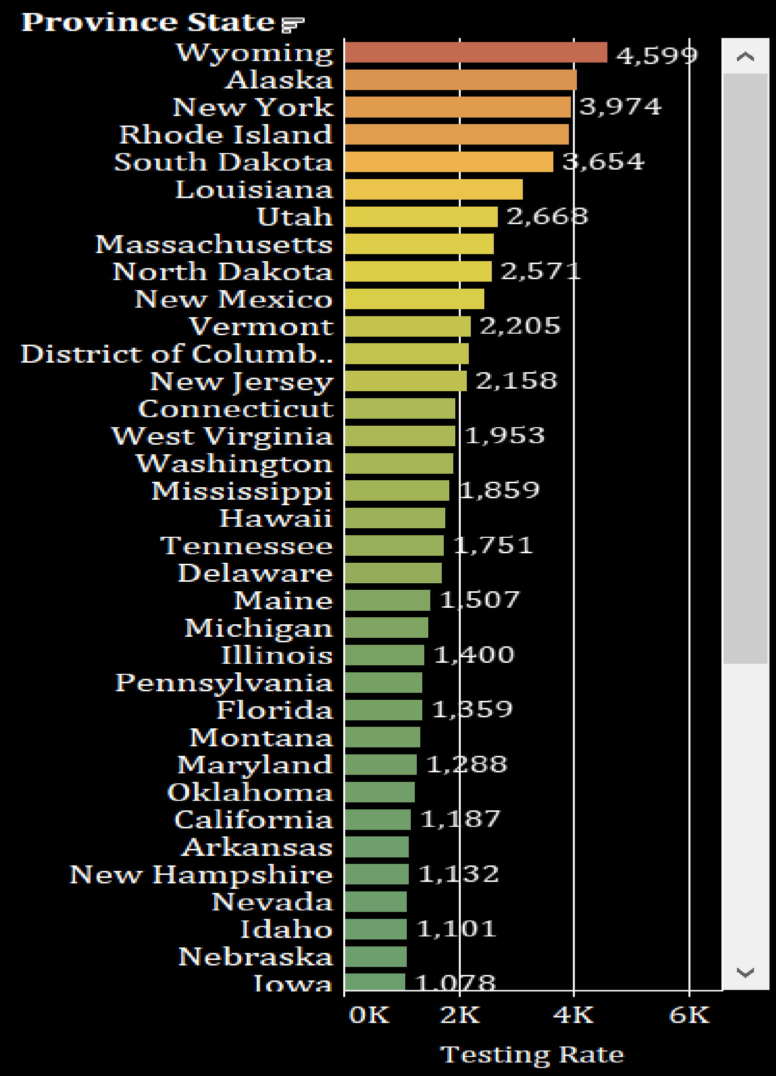


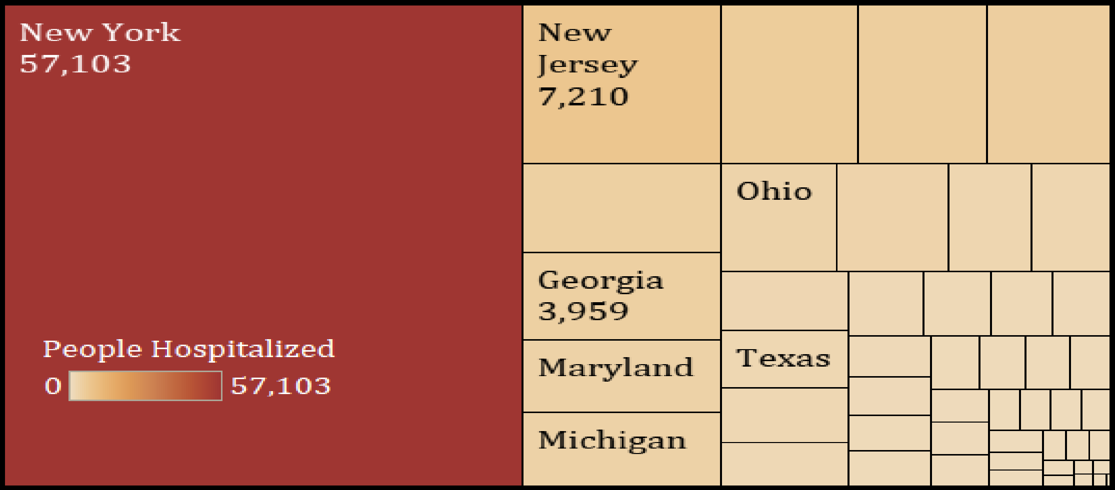
Province State	Confirmed..	Deaths..	Recovered..	Active	Fips	Incident Rate	People Tested
Alabama	5593	196	0	5397	1	119.28	48760
Alaska	335	9	196	326	2	56.04	12159
Arizona	5473	231	1265	5242	4	75.19	56601
Arkansas	2276	42	863	2234	5	87.91	29713
California	37344	1421	0	35923	6	95.24	465327
Colorado	10891	506	0	10385	8	192.19	48704
Connecticut	22469	1544	0	20925	9	630.22	69918
Delaware	3200	89	599	3111	10	328.62	16553
District of Columbia	3206	127	645	3079	11	454.27	15502
Florida	28309	893	0	27416	12	133.33	288627

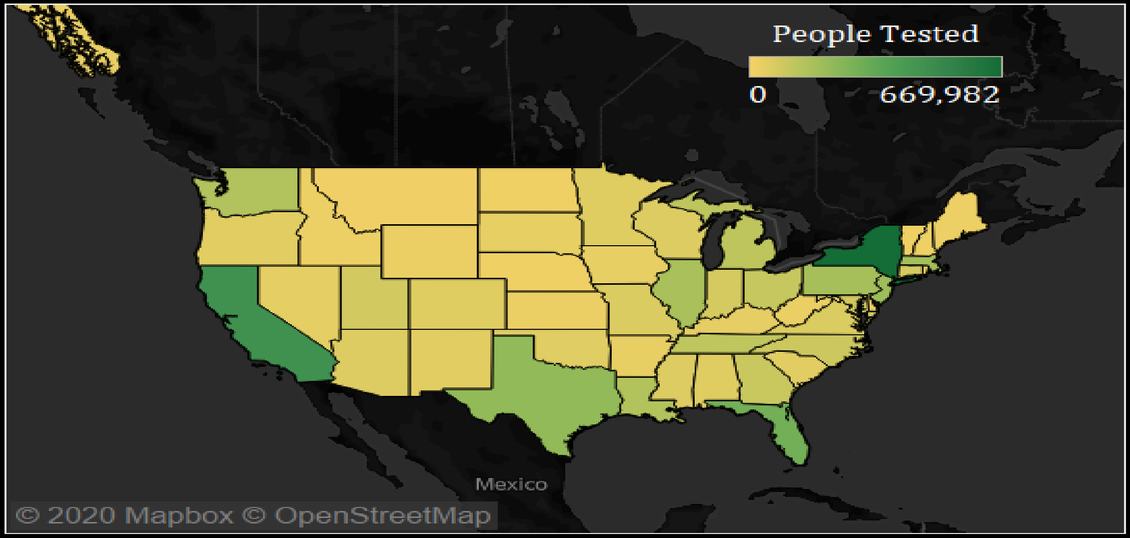

Canada
Deaths Density
1
100
200
300
400
United States
© 2020 Mapbox © OpenStreetMap
Mexico


Confirmed Cases
Confirmed
10K
5K
0K
Mar 2
Mar 7
Mar 12
Mar 17
Date [March 2020]


Active
0
244K
New York
243,879
Illinois
New Jersey
90,764


Province State
Wyoming 4,599
Alaska
New York 3,974
Rhode Island
South Dakota 3,654
Louisiana
Utah 2,668
Massachusetts
North Dakota 2,571
New Mexico
Vermont 2,205
District of Columb..
New Jersey 2,158
Connecticut
West Virginia 1,953
Washington
Mississippi 1,859
Hawaii
Tennessee 1,751
Delaware
Maine 1,507
Michigan
Illinois 1,400
Pennsylvania
Florida 1,359
Montana
Maryland 1,288
Oklahoma
California 1,187
Arkansas
New Hampshire 1,132
Nevada
Idaho 1,101
Nebraska
Iowa 1,078
0K
2K
4K
6K
Testing Rate

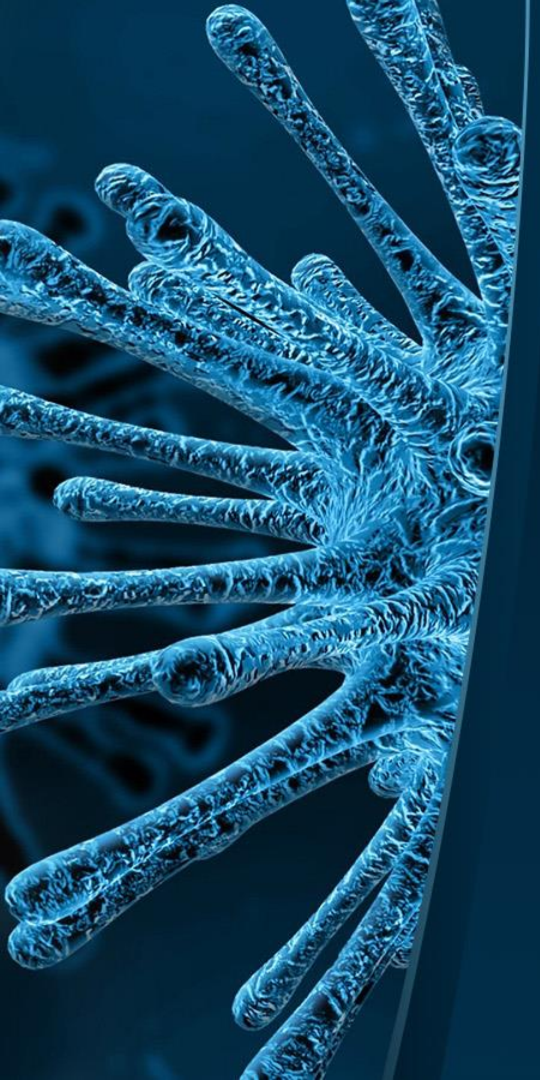

New York
57,103
New Jersey
7,210
Ohio
Georgia
3,959
Maryland
Texas
Michigan
People Hospitalized
0
57,103

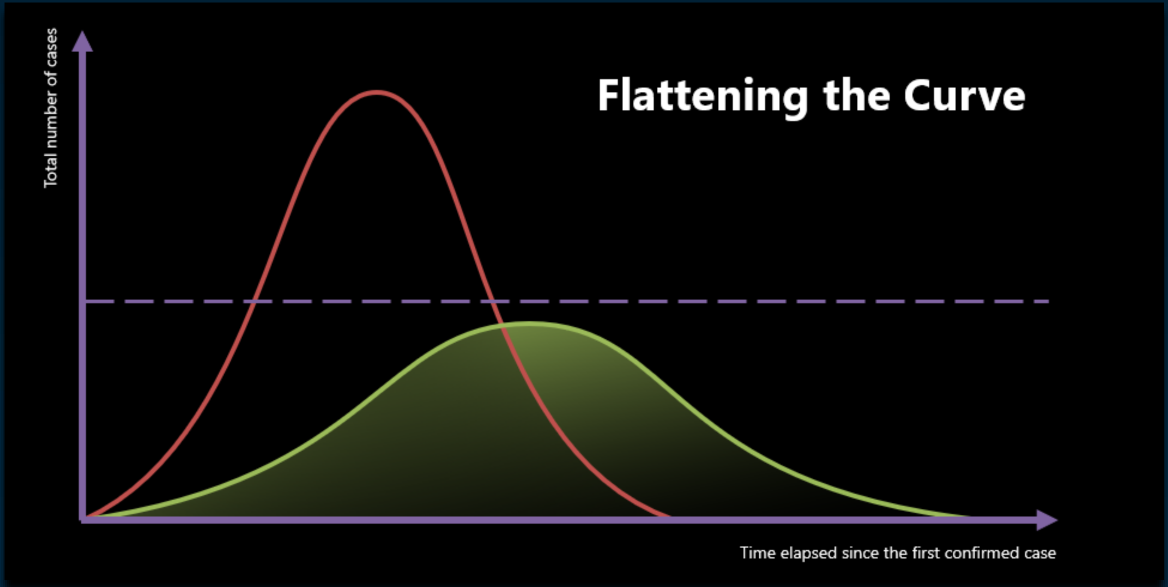

People Tested
0
669,982
Mexico
© 2020 Mapbox © OpenStreetMap

Summary

- Provided analysis of COVID19 pandemic to determine any relationships or trends.
- Followed shortened Agile framework for project schedule using checkpoints.
- Built pipeline that streams COVID-19 and provides a visualization.
- Selected GBM Multinomial model based AutoML recommendation leader using R-Squared, RMSE, and MSE.
- Designed Tableau dashboard to display COVID-19 pandemic data.

Any Questions or Comments?





Thank you