COVID-19 Final Report

Project Proposal

Abstract

Design and build a streaming application that provides a model, predictions, and visualization of COVID19 virus data in a Shiny dashboard. The objective of our dashboard is implementing a predictive model to identify which factors have the greatest impact on the spread and strength of the COVID-19 virus in a specific geographic area.

Project Objective

- Build a functional dashboard in Shiny presenting COVID19 virus predictions.
- Stream COVID data using Kafka or Spark Streaming.
- Visualization of predictive analytics model and data.
- Predictions or time series analysis based on COVID19, environmental, and economic factors.

Current Methods

Today, the most widely used predictive modeling techniques are decision trees, regression and neural networks. Regression (linear and logistic) is one of the most popular methods for statistical analysis. Another method used today is regression analysis which estimates relationships among variables. With models of the number of confirmed cases beginning to be released to the public, analysis on this data has become a top priority for Universities as well as Government Agencies. The limitations of the models available don't quantify the relationship between demographics in a populous (Other than identifying risk factors on an individual basis) and government actions on the extent and severity of the virus.

Our Approach

Most models relate only to the number of cases per geographic location or the number of hospitalization and deaths in a given area. There are also only general descriptive statistics on the risk factors related to the virus. This project would attempt to quantify the effect of government action, demographic data, healthcare resources and the population's compliance with government orders on the rate of spread and severity of health impact. Rather than simply stating risk factors as descriptive statistics, this analysis would attempt to provide predictive models based on the risk factors identified, as they appear in a given population.

Impact

Governments, healthcare workers, the general public, in short everyone has a stake in this information. The more we understand about this virus, the better we can be prepared when it comes back. With COVID19 and Coronavirus dominating the news and media, streaming virus data will provide a predictive analytics tool for tracking the impact of the pandemic. Creating awareness of the spread of the virus could help us better understand its impact. These results could help save lives.

Risks

The primary risks include incorrectly analyzing the data and providing misleading results. Incorrect conclusions could lead to damaging actions. Another risk is if states inaccurately identify confirmed cases or understate the total number of deaths. A payoff is correctly identifying the results that lead to taking actions that mitigate the adverse effects of the virus. If we can identify which actions lead to increased mitigation, then results could provide more hope and encouragement than fear.

Costs

The data is publicly available, and the tools being used in the analysis are open source including: Spark, R, Python, etc.

Checkpoints

- **Midpoint Checkpoint**: Feature and model selection, tuning, and validation.
- **Final Exam Checkpoint**: Dashboard functionality includes visualization, model results, and streaming data.

Initial data gathering and analysis to determine a statistically significant relationship will take a week. Attempts to most accurately model the relationship at a more granular level will take a week. Midterm determination would involve identifying a significant relationship and final examination would be determined by quantifying and modeling that given relationship with more granular detail. The Final Exam Checkpoint will focus primarily on testing and application deployment.

Project Schedule

Our project schedule is set for a 4-week deadline where we will implement a shortened agile process of 2 (1) week iterations, followed by (1) week of testing, and the final week is dashboard deployment.

Team

Ishaan Khurana: **Product Manager**

Jacqueline Connolly: **Pipeline**

Kyle W. Brown – **Machine Learning/Dashboard**

Derick Karolak – **Research/Support**

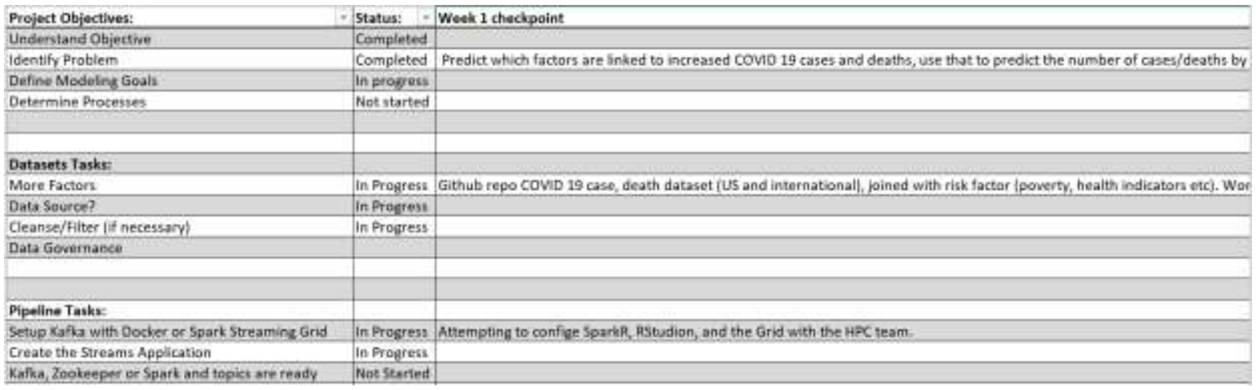
Nishchitha Manjunath – **Visualization**

Data Source

Link to public API's: https://rapidapi.com/collection/coronavirus-covid-19**.** Link to public description of data source: https://github.com/CSSEGISandData/COVID-19. Link to public of data source: https://github.com/CSSEGISandData/COVID-19/tree/master/csse_covid_19_data/csse_covid_19_daily_reports

Agile Schedule

The Agile Schedule was created from the initial task list and collaborated on. The original Agile Schedule Excel workbook with screenshot:



Project Objectives:	**Status:**	**Week 1 checkpoint**
Understand Objective	Completed	
Identify Problem	Completed	Predict which factors are linked to increased COVID 19 cases and deaths, use that to predict the number of cases/deaths by
Define Modeling Goals	In progress	
Determine Processes	Not started	
Datasets Tasks:		
More Factors	In Progress	Github repo COVID 19 case, death dataset (US and international), joined with risk factor (poverty, health indicators etc). Wor
Data Source?	In Progress	
Cleanse/Filter (if necessary)	In Progress	
Data Governance		
Pipeline Tasks:		
Setup Kafka with Docker or Spark Streaming Grid	In Progress	Attempting to confige SparkR, RStudion, and the Grid with the HPC team.
Create the Streams Application	In Progress	
Kafka, Zookeeper or Spark and topics are ready	Not Started	

Jacqueline Connolly: Pipeline

Introduction

When attempting to build a data science application that displays results to end users and works from changing and evolving data, it is important to build a pipeline to ensure that the application stays up to date. There are many steps involved in building a pipeline that can accomplish this task. The data is available at the source in a certain format and having a certain quality, and this data will need to be processed to varying degrees before it can be analyzed. Once the data has been cleaned and analyzed, the results can be displayed within the application for consumption by the end user.

There are several tools that can be used in building a pipeline, all of which have different advantages and disadvantages. It is important to note that when creating an application that incorporates a variety of models, a variety of branches to the pipeline will also be needed. While the mental image that results from the term pipeline, gives the impression of a single tube connecting one point to another with maybe some twists and turns in between, actual pipelines for more complex applications have a structure more similar to a tree; with data starting at the leaves, different cleaning and analytics processing existing along the forks in the branches, and it all coming together at the base where the final application would be. Robust pipelines also allow for flexibility and change. Often in science, attempting to answer one question has the result of generating more questions. The ability for a pipeline to adjust and adapt as projects evolve without bringing down existing applications is an important consideration when constructing a pipeline.

Approach

The general approach was to build a robust and flexible pipeline that would allow for future improvement, expansion, and model incorporation, one step at a time, starting at the data source and finishing at the end application. Completely automating every step of the process was important to produce a result where no human intervention would be needed for the application to stay current and up to date.

The goal was to build a base pipeline that would serve as a template and starting point from which more advanced analytics could then be incorporated and thus more meaningful results then displayed in the end application.

Tools

Python was the primary language implemented in this pipeline. Python is a high-level programming language with a multitude of user-friendly libraries whose ease-of-use allow for rapid prototyping. When attempting to put together an application that utilizes multiple data sources and integrates with other software, python has libraries that allow for such integrations with only a few lines of code. Python also has powerful libraries such as numpy and pandas that make data cleaning and data processing quick and simple. Because it is such a high-level language, it is not the most efficient of languages when creating a production-level pipeline where speed and efficiency are key. For this purpose, another more performant language such as Scala, would be recommended.

Kafka is a powerful tool that allows for high levels of flexibility within pipelines. It's publish/subscribe structure allows for the decoupling of producers and consumers, making it possible to create pipelines where many sources feed into a single topic to then be consumed as one data source by the next process or processes and/or to have one source feed to a topic that can then be consumed by many different processes for many different applications. Kafka helps with fault tolerance and provides a buffer when applications are unable to process data at the rate by which the data is being produced. It allows for horizontal scalability because it is distributed in nature. It also increases pipeline robustness by adding modularity so that pieces of the pipeline may be added and removed with minimal effect on the rest of the pipeline. For our purposes Kafka was a useful tool in organizing the pipeline so that it may be expanded upon in the future to feed multiple models with varying degrees of data processing and transformation.

D3 was selected as the endpoint for this base pipeline due to its asynchronous qualities which are useful in building a dashboard that experiences frequent updates. The ability of D3 visualizations to update when the underlying data changes without the page needing to be refreshed has important implications for dashboards displayed in public settings such as businesses or organizations, where there may not be an individual directly interacting with the application and able to refresh as needed.

Implementation

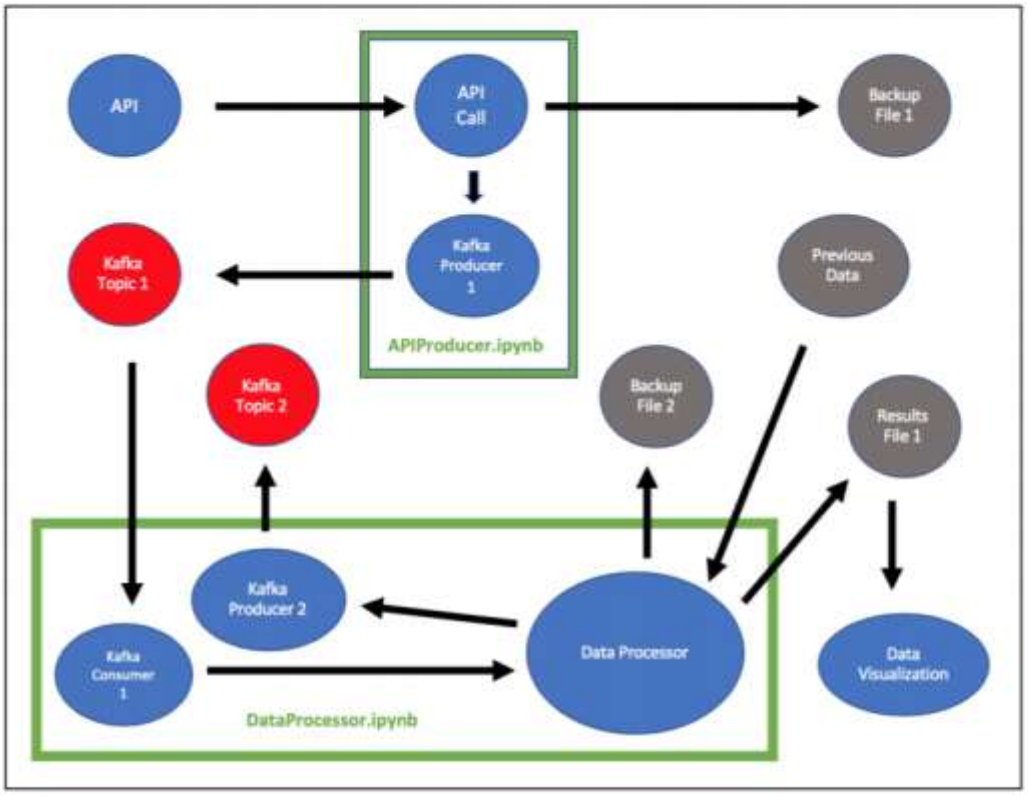
For this project we chose to analyze COVID-19 data to produce results that may be meaningful to the general public in understanding the crisis that surrounds us. The final application was decided to focus on visualizations that make the results more consumable to the target audience. Different models were assessed to produce the results that would be displayed. A pipeline was needed to move the data from source, through processing and cleaning, to the model for analysis, and then to move the results to the end application which would display the visualizations to the user and/or target audience. Our primary source for static and historic data was the John Hopkins GitHub repository.

This data was available for download as a text file and was subsequently available for import into the pipeline from the local filesystem. The dynamic and evolving data that would allow our application to stay current and up to date was available through a free API which could be called periodically through code to feed into the pipeline. The API calls made available a range of variables, only some of which were used in analysis. The API provided; New Cases, Active Cases, Critical Cases, Recovered Cases, Total Cases, New Deaths, Total Deaths, and Total Tests by country. A script was needed to isolate the data that would be used in analysis. For this base pipeline, only the number of total cases were needed.

After isolating the relevant data from the API response, the data still needed to be cleaned before it could be combined with the historic data from JHU. Some of the country names were formatted differently between the two data sources. Some of the API names had hyphens in them where the JHU set had spaces, some countries were abbreviated while others were not (United Arab Emirates vs UAE) and some countries were abbreviated differently, such as 'US' in one dataset compared to 'USA' in the other.

After the cleaning was performed, the datasets were ready to be combined. The number of cases on January 22nd, the earliest recorded cases from the JHU dataset, were combined with the most recent totals of cases from the API data.

After the data was prepared for analysis, it was ready to be fed into the model. For this base pipeline, analysis meant sorting the number of total cases to identify the top 10 countries with the most cases at the current point in time. Once results were produced, they were consumed by end application where the data was visualized and displayed for the user. Below is a flow chart of the base pipeline created.



In order to run the pipeline, zookeeper must be started, then the kafka broker, a local server needs to be running to make the final results accessible to D3, and the python kernel must also be running. The python scripts need to be started to begin moving data. The local server port needs to be accessed in the browser to make the local files available to D3, then the HTML page housing the end application may be opened. Once these steps are complete, no additional intervention is needed for the dashboard to display the most current data. Screen shots of specific code are available in the appendix as well as a snapshot of the D3 endpoint.

Next Steps

To achieve the objective of creating an application that utilizes current data, performs meaningful analysis and displays consumable visualizations of the results to the target audience, meaningful analytics models must be incorporated into the base pipeline. In order to incorporate these more advanced models, additional cleaning scripts need to be created to ensure proper formatting of the data for the model by which it is to be consumed. The code for the models must be adjusted to include consumers, so that they are able to read the up to date, cleaned data for processing. The model script must also be wrapped in an infinite loop with a timer so that it is able to update the results periodically. Finally, the result visualizations must be consolidated into a single platform that can dynamically read the results from the models as they are updated and display the changes for the audience.

Challenges

Due to the diversity of tools and methods used in analysis, in order to complete the next steps, it would be necessary for the individuals who created the models to isolate the steps they performed in the cleaning of the data and to consolidate these steps into a separate script or document so that the data cleaning process for each of the models could then be analyzed to identify similarities and differences in the needs of the respective models. Redundancies could then be eliminated by combining the shared processing steps into a single optimized script. The generally processed data would then be written to a topic for shared consumption. Smaller scripts could then be written with only the steps unique to that individual model for further processing. It is possible that some of the smaller scripts may not be necessary and could potentially be combined with the model script to eliminate adding an unnecessary stage in the pipeline.

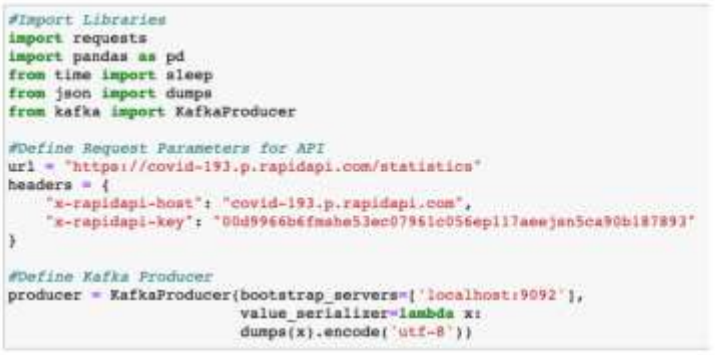
For efficiency and consistency, it would be necessary to try to identify the furthest common point in data processing before splitting the pipeline. Another challenge in completing the pipeline is the variety of platforms considered for the application endpoint. The original endpoint was supposed to be shiny but that decision was changed and some work began with tableau. D3 was utilized for the base pipeline endpoint for the reasons stated in the Tools section above. The visualizations present in the differing endpoints do not currently correspond to the output generated by the models developed. In order to finish the pipeline for an application that achieves the project objectives, a single platform would need to be decided upon for the endpoint, and visualizations relative to model outcomes would need to be developed.

Conclusion

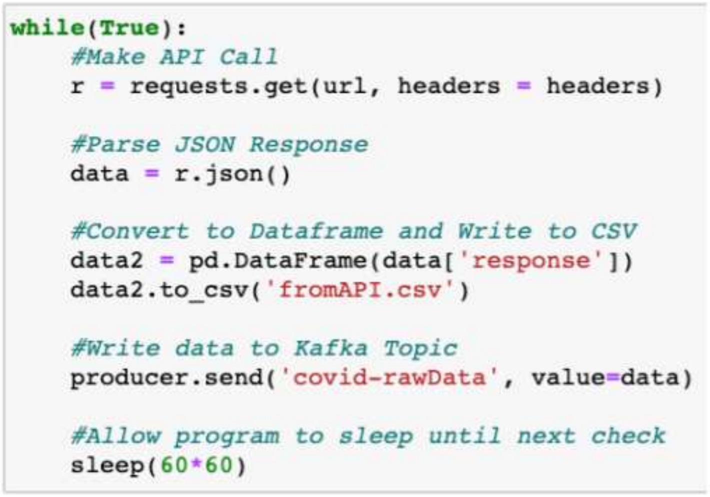
A base pipeline was built to provide a starting point and template to be expanded upon until reaching the project objectives. Data sources were accessed and incorporated into the pipeline. Initial cleaning scripts were created. Kafka topics were created to increase modularity and to allow for flexibility as the more advanced analytics models were incorporated into the pipeline. Kafka producers and consumers were utilized as entry and exit points to the different pipeline stages. A dynamic visualization was created as a stand-in end point until the more advanced analytics and appropriate visualizations could be integrated. Intermediate results were also written to text files for additional persistence and fault tolerance.

While some of the tools used for analytics in this project may not be able to directly utilize a Kafka consumer or producer, all data analytics tools can read in text files, so if needed the pipeline would be able to incorporate those models by writing the cleaned data to plain text or csv for model consumption and automating the updating of the model by creating a shell script to run the model periodically. If the model can write the results to text file or csv then the pipeline would be able to pick the results back up if any additional processing was needed or the end application could read the results directly from the file. While we were unable to finish a pipeline for an application that meets the original objectives, a fully automated pipeline utilizing COVID-19 data sources was created as a template and starting point for additional improvement and expansion.

APIProducer.ipynb

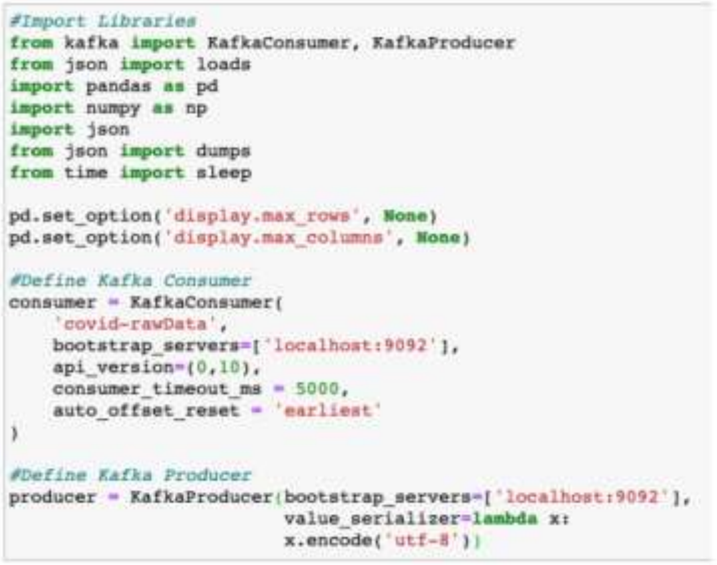


```
#Import Libraries
import requests
import pandas as pd
from time import sleep
from json import dumps
from kafka import KafkaProducer

#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'))
```

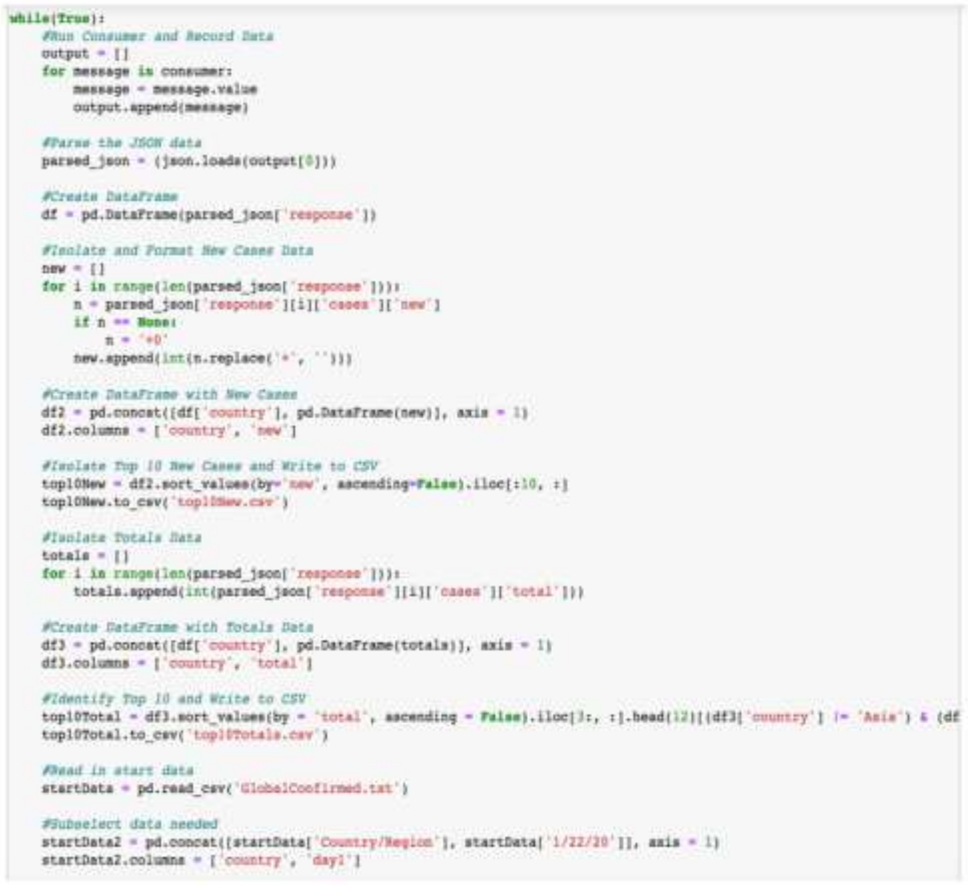


```
while(True):
    #Make API Call
    r = requests.get(url, headers = headers)

    #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)
```

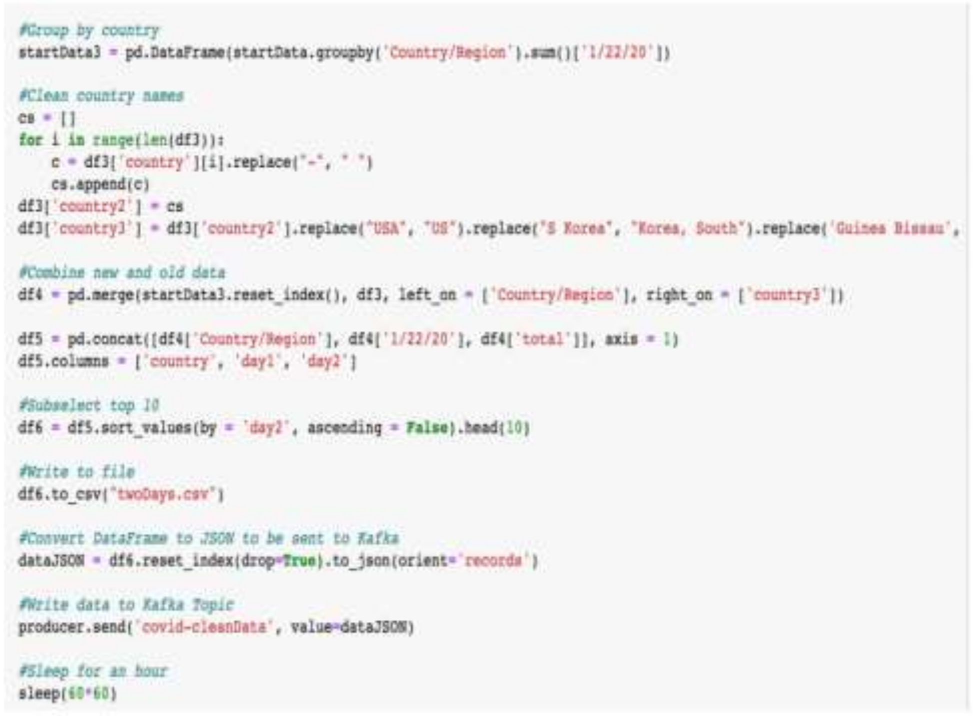
DataProcessor.ipynb



```
#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'))
```

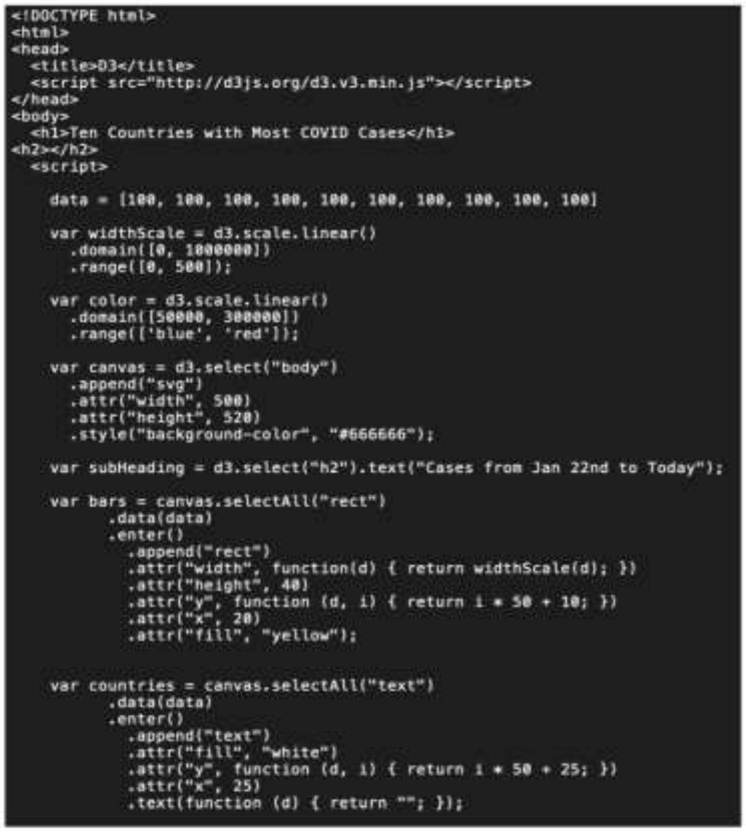


```
while(True):
    #Run Consumer and Record Data
    output = []
    for message in consumer:
        message = message.value
        output.append(message)

    #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']
```

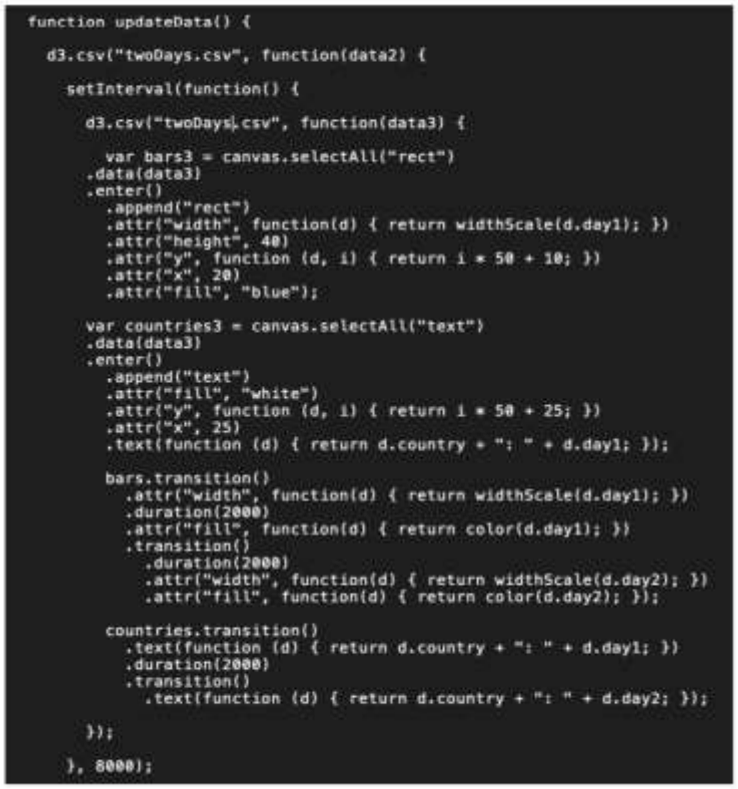


```
#Group by country
startData3 = pd.DataFrame(startData.groupby('Country/Region').sum()['1/22/20'])

#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)
```

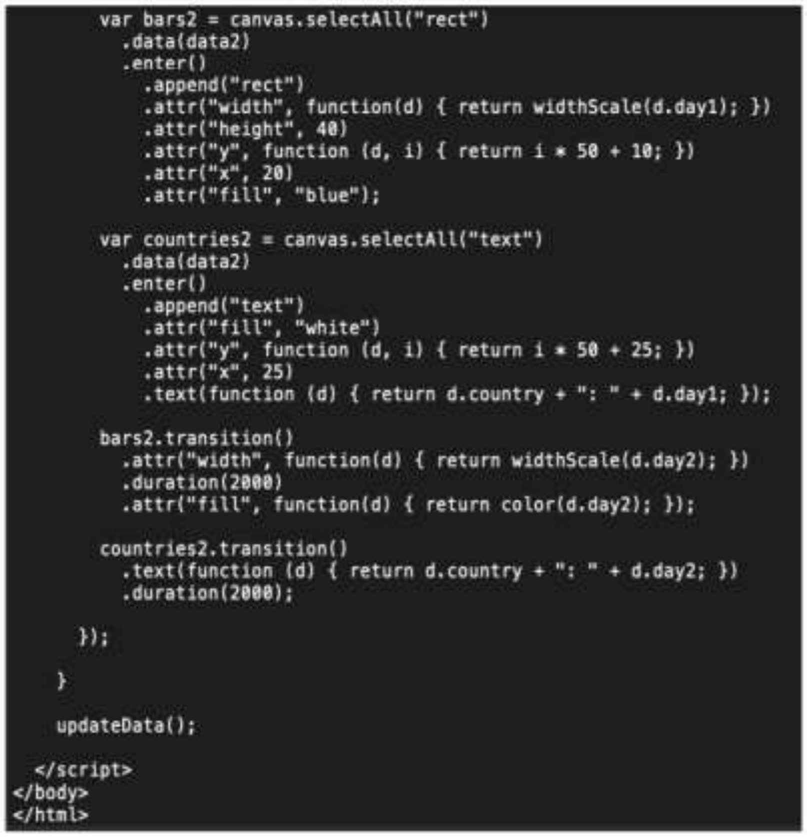
covidVisualizations.html



```
<!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 ""; });
```

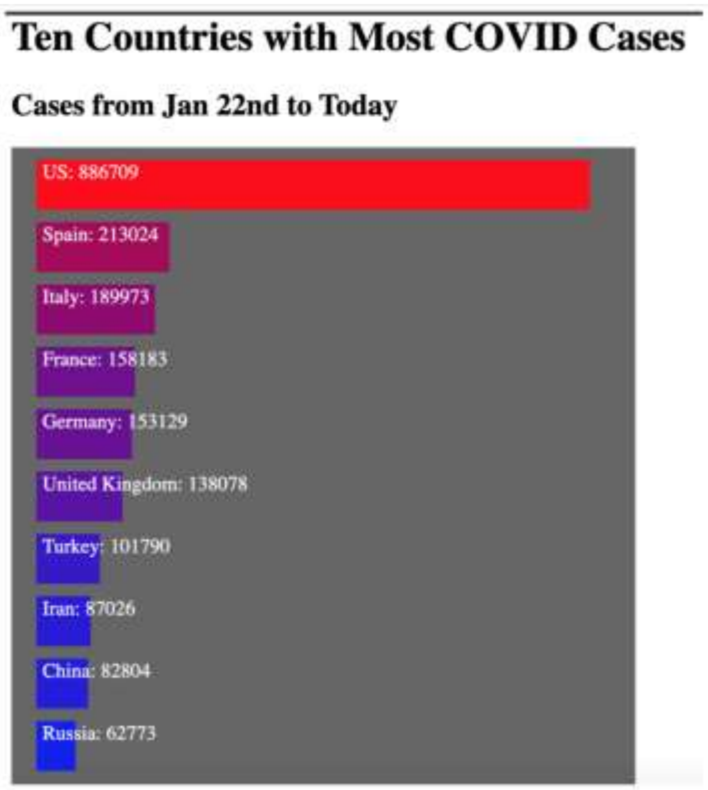


```
  function updateData() {

    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);
```



```
        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>
```

Snapshot of Endpoint Webpage



Ten Countries with Most COVID Cases

Cases from Jan 22nd to Today

Derick Karolak: Research/Support

Checkpoint 1

1. This week I worked on:
   a. Data discovery
   b. Fitting the data into different Models
   c. Identified statistically significant relationships within the data set

2. Next week I plan on working on getting the data into a database that has a live connection to the data source to code against.

3. I am currently still struggling on getting my IntelliJ environment setup and configured correctly. Which has been a struggle since week 6.

Checkpoint 2

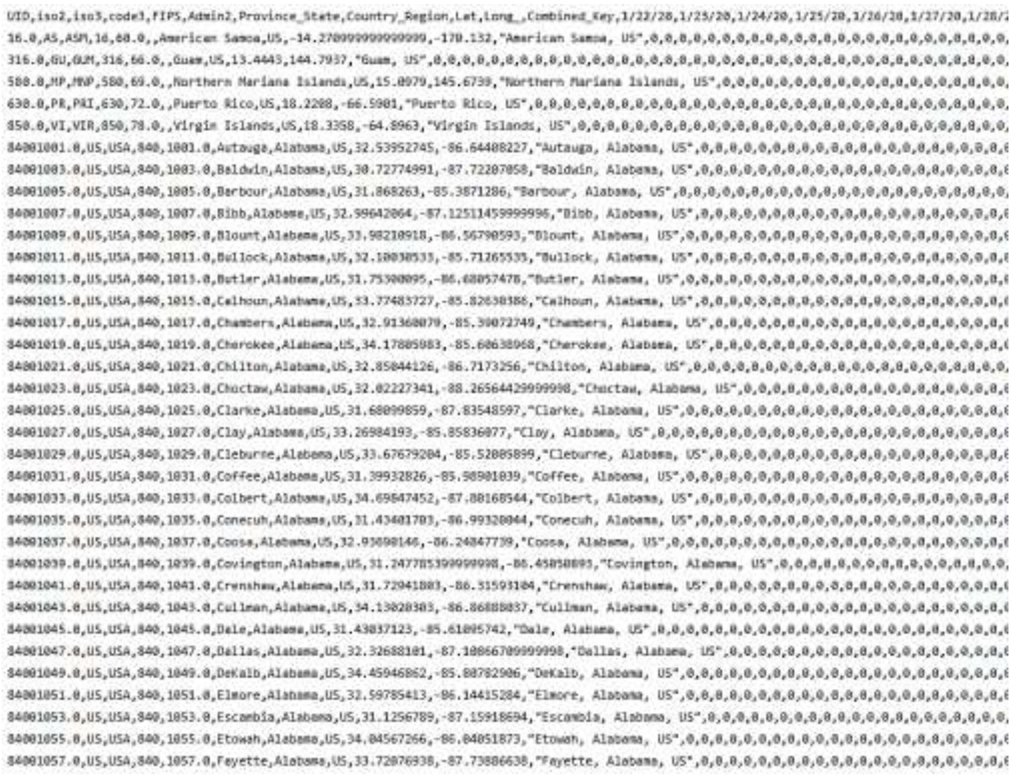
1. What did you do this week?
   - Continued to try to run regression models but kept failing to get anything that made sense.
   - Started working on the data pipeline from the source to the grid.
2. What are you planning to do next week?
   - Complete the data pipeline
   - Work on the power point presentation
   - Work on the report
3. What problems are you having?
   - Still plagued by issues in InteliJ and my lowered confidence as a result of my continued struggles.

Ishaan Khurana: Product Manager

Statistical Question
When and where will the peak of coronavirus cases occur? At which date will the number of total cases stop increasing significantly and in which countries/ counties will it occur first?

Data collection and preparation
The only type of model that can use historical data to predict future values is time series. For time series models, the input data required to predict the future number of total cases is historical case data for each day (containing location data).

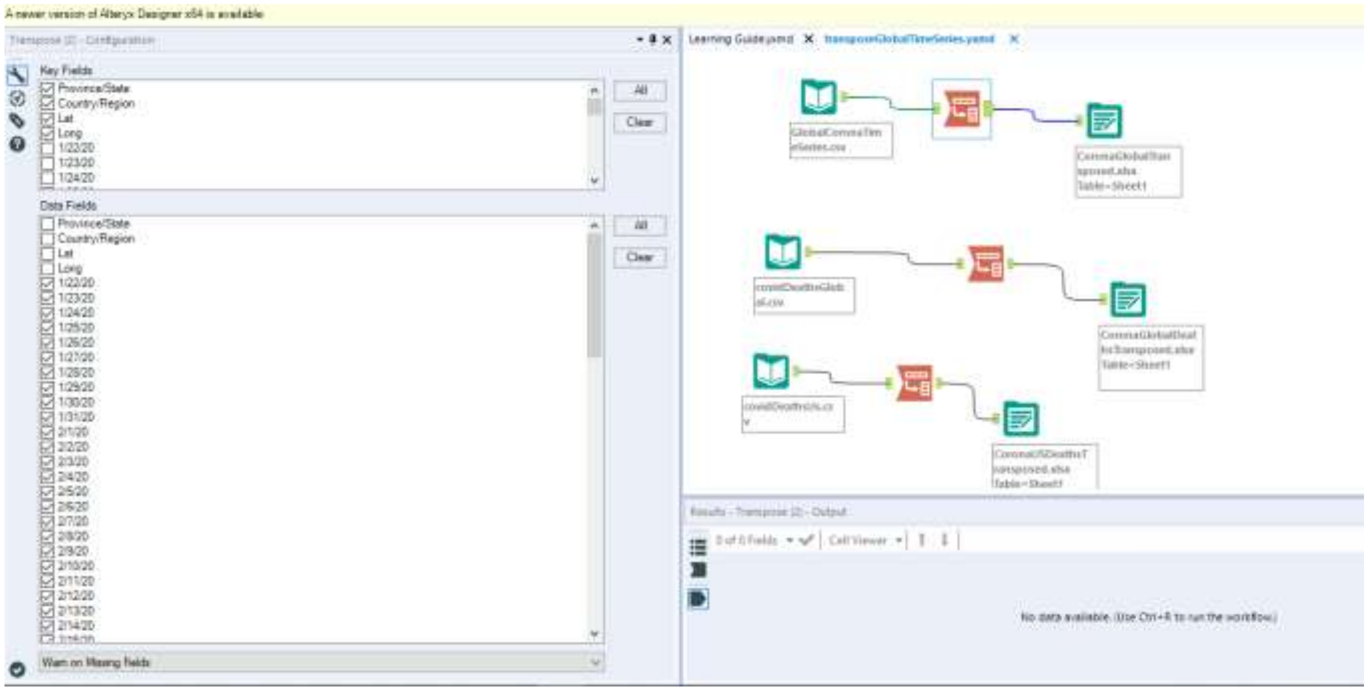
Raw time series data from John Hopkins is appended daily with new case data for US and world:



```
UID,iso2,iso3,code3,FIPS,Admin2,Province_State,Country_Region,Lat,Long_,Combined_Key,1/22/20,1/23/20,1/24/20,1/25/20,1/26/20,1/27/20,1/28/2
16.0,AS,ASM,16,60.0,,American Samoa,US,-14.270999999999999,-170.132,"American Samoa, US",0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
316.0,GU,GUM,316,66.0,,Guam,US,13.4443,144.7937,"Guam, US",0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
580.0,MP,MNP,580,69.0,,Northern Mariana Islands,US,15.0979,145.6739,"Northern Mariana Islands, US",0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
630.0,PR,PRI,630,72.0,,Puerto Rico,US,18.2208,-66.5901,"Puerto Rico, US",0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
850.0,VI,VIR,850,78.0,,Virgin Islands,US,18.3358,-64.8963,"Virgin Islands, US",0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
84001001.0,US,USA,840,1001.0,Autauga,Alabama,US,32.53952745,-86.64408227,"Autauga, Alabama, US",0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
84001003.0,US,USA,840,1003.0,Baldwin,Alabama,US,30.72774991,-87.72207058,"Baldwin, Alabama, US",0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
84001005.0,US,USA,840,1005.0,Barbour,Alabama,US,31.868263,-85.3871286,"Barbour, Alabama, US",0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
84001007.0,US,USA,840,1007.0,Bibb,Alabama,US,32.99642064,-87.12511459999996,"Bibb, Alabama, US",0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
84001009.0,US,USA,840,1009.0,Blount,Alabama,US,33.98210918,-86.56790593,"Blount, Alabama, US",0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
84001011.0,US,USA,840,1011.0,Bullock,Alabama,US,32.10030533,-85.71265535,"Bullock, Alabama, US",0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
84001013.0,US,USA,840,1013.0,Butler,Alabama,US,31.75300095,-86.68057478,"Butler, Alabama, US",0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
84001015.0,US,USA,840,1015.0,Calhoun,Alabama,US,33.77483727,-85.82630386,"Calhoun, Alabama, US",0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
84001017.0,US,USA,840,1017.0,Chambers,Alabama,US,32.91360079,-85.39072749,"Chambers, Alabama, US",0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
84001019.0,US,USA,840,1019.0,Cherokee,Alabama,US,34.17805983,-85.60638968,"Cherokee, Alabama, US",0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
84001021.0,US,USA,840,1021.0,Chilton,Alabama,US,32.85044126,-86.7173256,"Chilton, Alabama, US",0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
84001023.0,US,USA,840,1023.0,Choctaw,Alabama,US,32.02227341,-88.26564429999998,"Choctaw, Alabama, US",0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
84001025.0,US,USA,840,1025.0,Clarke,Alabama,US,31.68099859,-87.83548597,"Clarke, Alabama, US",0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
84001027.0,US,USA,840,1027.0,Clay,Alabama,US,33.26984193,-85.85836077,"Clay, Alabama, US",0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
84001029.0,US,USA,840,1029.0,Cleburne,Alabama,US,33.67679204,-85.52005899,"Cleburne, Alabama, US",0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
84001031.0,US,USA,840,1031.0,Coffee,Alabama,US,31.39932826,-85.98901039,"Coffee, Alabama, US",0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
84001033.0,US,USA,840,1033.0,Colbert,Alabama,US,34.69847452,-87.80168544,"Colbert, Alabama, US",0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
84001035.0,US,USA,840,1035.0,Conecuh,Alabama,US,31.43401703,-86.99320044,"Conecuh, Alabama, US",0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
84001037.0,US,USA,840,1037.0,Coosa,Alabama,US,32.93090146,-86.24847739,"Coosa, Alabama, US",0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
84001039.0,US,USA,840,1039.0,Covington,Alabama,US,31.247785399999998,-86.45050893,"Covington, Alabama, US",0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
84001041.0,US,USA,840,1041.0,Crenshaw,Alabama,US,31.72941803,-86.31593104,"Crenshaw, Alabama, US",0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
84001043.0,US,USA,840,1043.0,Cullman,Alabama,US,34.13020303,-86.86888037,"Cullman, Alabama, US",0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
84001045.0,US,USA,840,1045.0,Dale,Alabama,US,31.43037123,-85.61095742,"Dale, Alabama, US",0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
84001047.0,US,USA,840,1047.0,Dallas,Alabama,US,32.32688101,-87.10866709999998,"Dallas, Alabama, US",0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
84001049.0,US,USA,840,1049.0,DeKalb,Alabama,US,34.45946862,-85.80782906,"DeKalb, Alabama, US",0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
84001051.0,US,USA,840,1051.0,Elmore,Alabama,US,32.59785413,-86.14415284,"Elmore, Alabama, US",0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
84001053.0,US,USA,840,1053.0,Escambia,Alabama,US,31.1256789,-87.15918694,"Escambia, Alabama, US",0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
84001055.0,US,USA,840,1055.0,Etowah,Alabama,US,34.04567266,-86.04051873,"Etowah, Alabama, US",0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
84001057.0,US,USA,840,1057.0,Fayette,Alabama,US,33.72076938,-87.73886638,"Fayette, Alabama, US",0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

Although the data is formatted correctly, and doesn't require cleaning, it must be transposed. The time series model requires either a date/time index by row (there are dates in the raw data but they are appended by column), or just the values by order of time with each row representing an equal time interval.

I used Alteryx transpose the data so that each day's values are represented by a row.



A newer version of Alteryx Designer x64 is available

Transpose (2) - Configuration

Key Fields
- ✓ Province/State
- ✓ Country/Region
- ✓ Lat
- ✓ Long
- ☐ 1/22/20
- ☐ 1/23/20
- ☐ 1/24/20

Data Fields
- ☐ Province/State
- ☐ Country/Region
- ☐ Lat
- ☐ Long
- ✓ 1/22/20
- ✓ 1/23/20
- ✓ 1/24/20
- ✓ 1/25/20
- ✓ 1/26/20
- ✓ 1/27/20
- ✓ 1/28/20
- ✓ 1/29/20
- ✓ 1/30/20
- ✓ 1/31/20
- ✓ 2/1/20
- ✓ 2/2/20
- ✓ 2/3/20
- ✓ 2/4/20
- ✓ 2/5/20
- ✓ 2/6/20
- ✓ 2/7/20
- ✓ 2/8/20
- ✓ 2/9/20
- ✓ 2/10/20
- ✓ 2/11/20
- ✓ 2/12/20
- ✓ 2/13/20
- ✓ 2/14/20

Warn on Missing Fields

Learning Guide.yxmd | transposeGlobalTimeSeries.yxmd

GlobalCoronaTimeSeries.csv → CoronaGlobalTransposed.xlsx Table=Sheet1

covidDeathsGlobal.csv → CoronaGlobalDeathsTransposed.xlsx Table=Sheet1

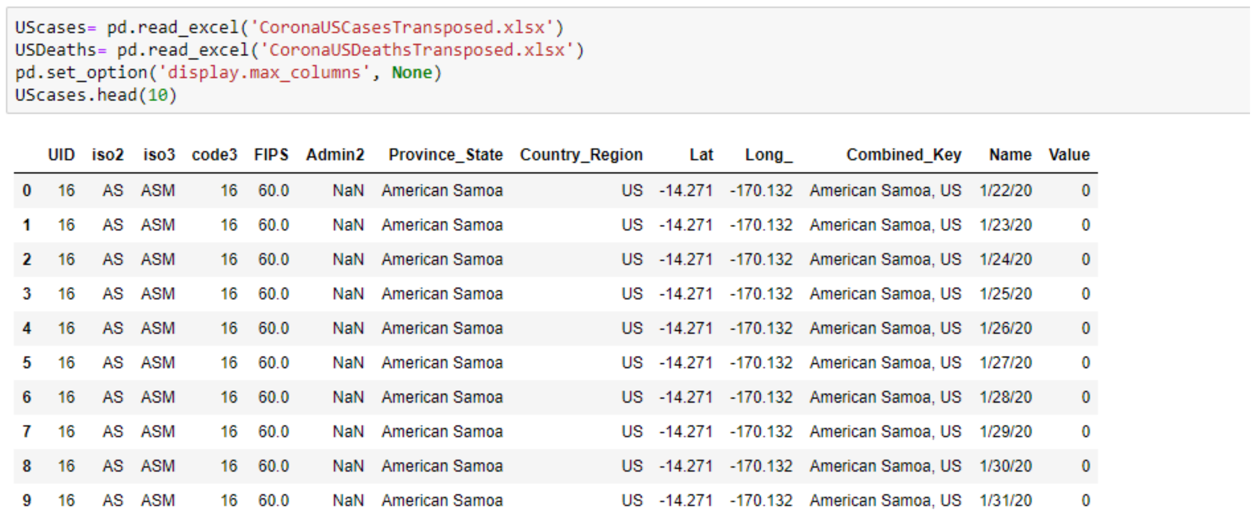
covidDeathsUS.csv → CoronaUSDeathsTransposed.xlsx Table=Sheet1

Results - Transpose (2) - Output

No data available. (Use Ctrl+R to run the workflow.)

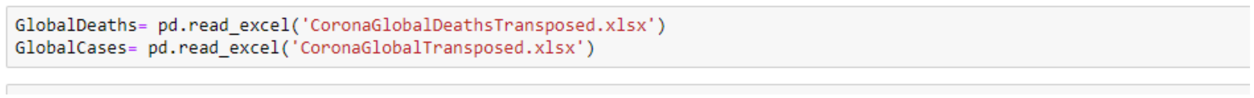
At this point, the data was ready to be inputted into Pandas Data frames for model development.

Model Development



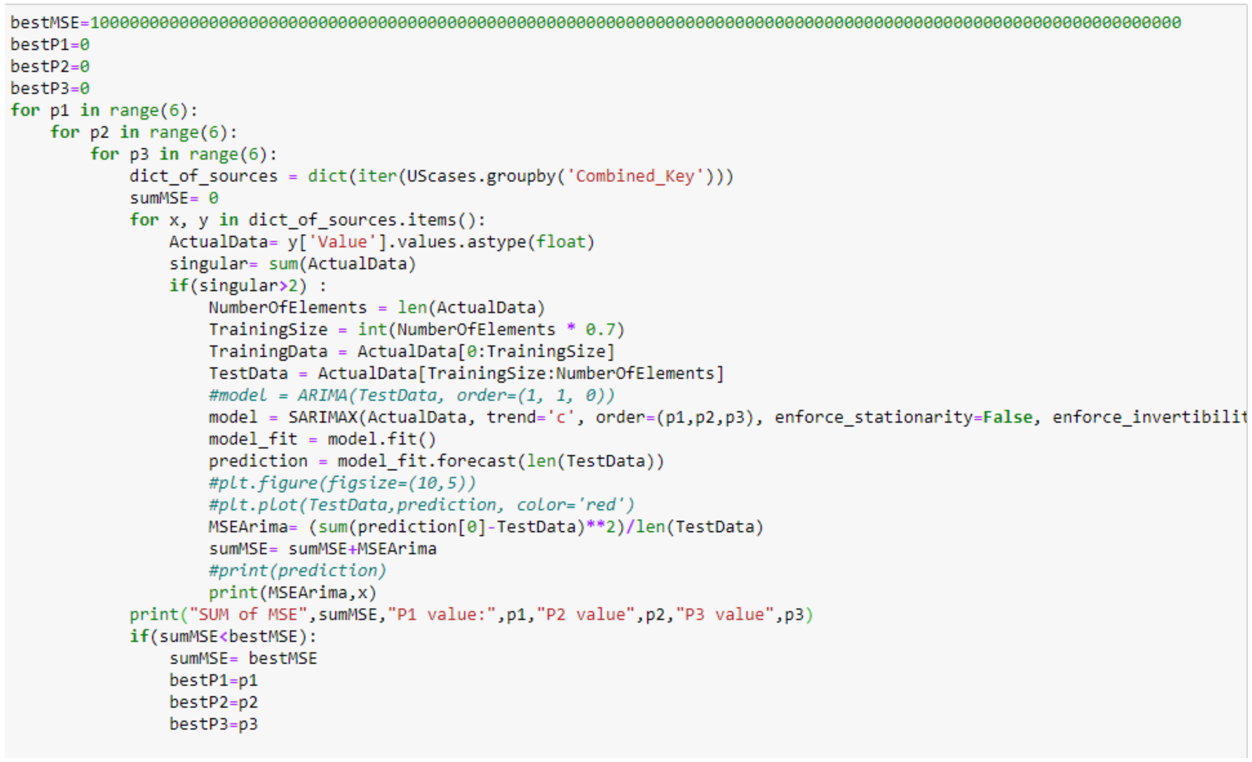
```
UScases= pd.read_excel('CoronaUSCasesTransposed.xlsx')
USDeaths= pd.read_excel('CoronaUSDeathsTransposed.xlsx')
pd.set_option('display.max_columns', None)
UScases.head(10)
```

	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



```
GlobalDeaths= pd.read_excel('CoronaGlobalDeathsTransposed.xlsx')
GlobalCases= pd.read_excel('CoronaGlobalTransposed.xlsx')
```

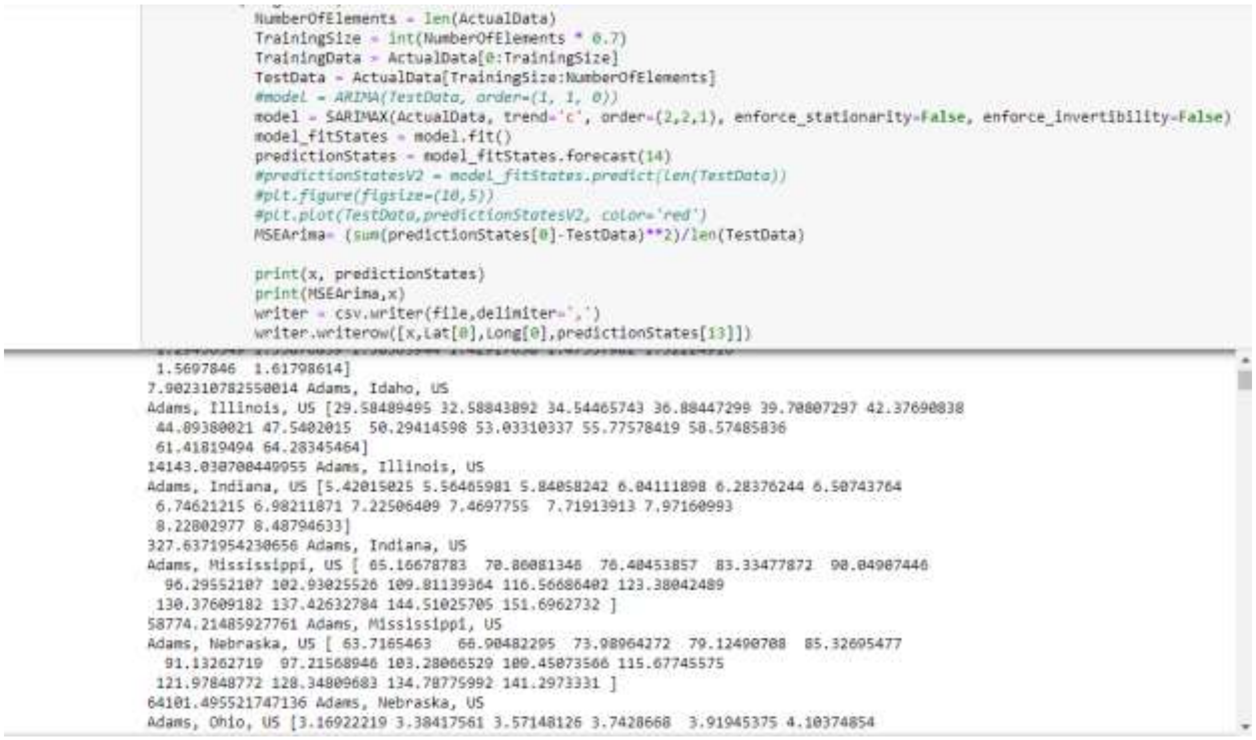
This is the loop I used to optimize the time series models fits.



```
bestMSE=1000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
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('Combined_Key')))
            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_invertibilit
                    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
```

First, the data is grouped by the combined key, which is the location (county or country), and moved into a dictionary. In the dictionary x is the combined location key, and y is the set of values that are the number of total cases per day from January to April. The first loop selects the first location and split the case count data into training/testing (70/30 split). It creates a SARIMAX time series model with base parameters (0,0,0) using the training data. Then it “predicts” the number of cases for the same dates as the test data (past dates). The predicted values are compared with the test data actual values and MSE is calculated for that combined key/location. This same process is repeated for each location, after all locations are created the total MSE across all is summed and saved. This same loop is repeated in a set of three other loops corresponding to each SARIMAX parameter p1, p2, p3 to find the combination that results in the lowest total MSE.

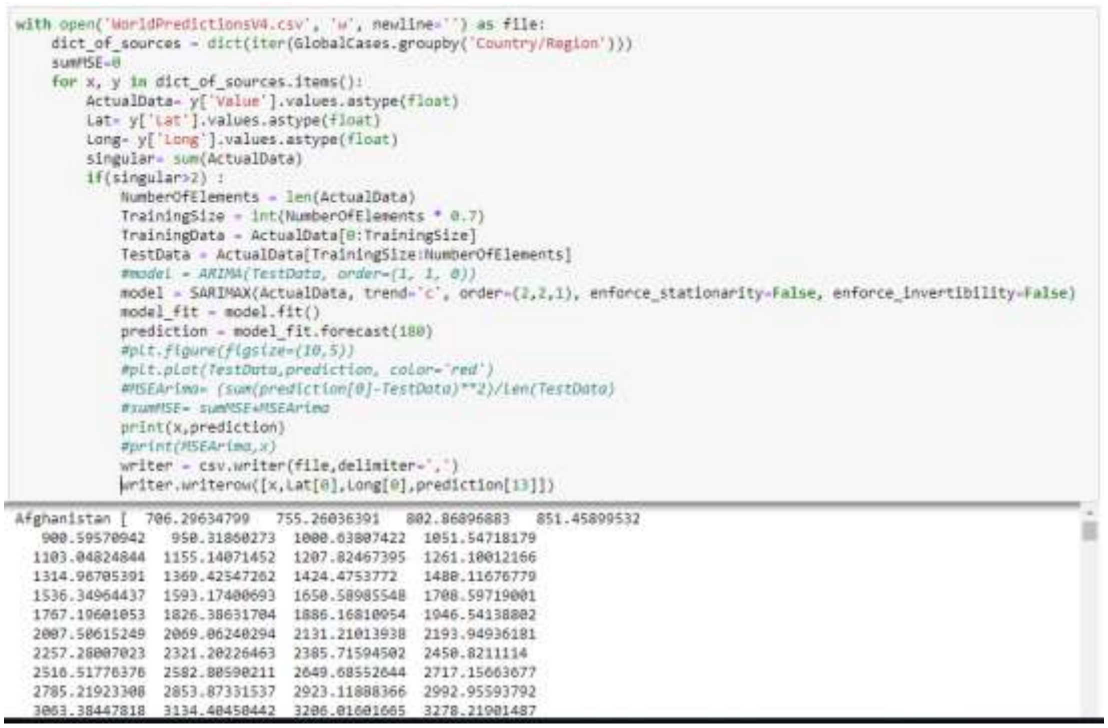
The best parameter combination for US county case data (2,2,1) is used to predict the number of cases for each county for dates extending after the historical dataset (future values).



```
            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=(2,2,1), enforce_stationarity=False, enforce_invertibility=False)
            model_fitStates = model.fit()
            predictionStates = model_fitStates.forecast(14)
            #predictionStatesV2 = model_fitStates.predict(len(TestData))
            #plt.figure(figsize=(10,5))
            #plt.plot(TestData,predictionStatesV2, color='red')
            MSEArima= (sum(predictionStates[0]-TestData)**2)/len(TestData)

            print(x, predictionStates)
            print(MSEArima,x)
            writer = csv.writer(file,delimiter=',')
            writer.writerow([x,Lat[0],Long[0],predictionStates[13]])
```

```
 1.5697846  1.61798614]
7.902310782550014 Adams, Idaho, US
Adams, Illinois, US [29.58489495 32.58843892 34.54465743 36.88447299 39.70807297 42.37690838
 44.09380021 47.5402015  50.29414598 53.03310337 55.77578419 58.57485836
 61.41819494 64.28345464]
14143.038700449955 Adams, Illinois, US
Adams, Indiana, US [5.42015025 5.56465981 5.84058242 6.04111898 6.28376244 6.50743764
 6.74621215 6.98211871 7.22506409 7.4697755  7.71913913 7.97160993
 8.22802977 8.48794633]
327.6371954230656 Adams, Indiana, US
Adams, Mississippi, US [ 65.16678783  70.86081346  76.40453857  83.33477872  90.04907446
  96.29552107 102.93025526 109.81139364 116.56686402 123.38042489
 130.37609182 137.42632784 144.51025705 151.6962732 ]
58774.21485927761 Adams, Mississippi, US
Adams, Nebraska, US [ 63.7165463   66.90482295  73.98964272  79.12490708  85.32695477
  91.13262719  97.21568946 103.28066529 109.45073566 115.67745575
 121.97848772 128.34809683 134.78775992 141.2973331 ]
64101.495521747136 Adams, Nebraska, US
Adams, Ohio, US [3.16922219 3.38417561 3.57148126 3.7428668  3.91945375 4.10374854
```

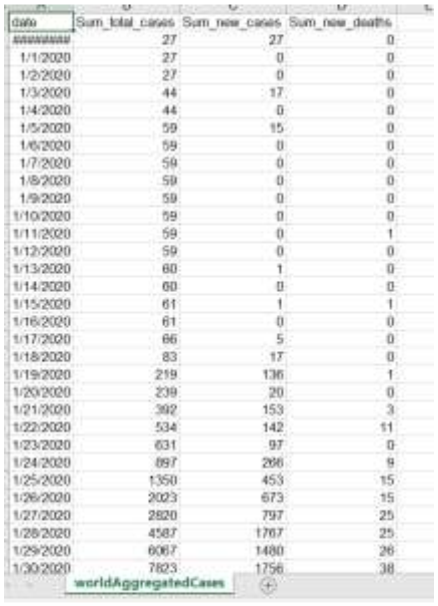
The same process is repeated for global cases by country. The most recent predictions are saved with latitude and longitude for visualizations in Tableau.



```
with open('WorldPredictionsV4.csv', 'w', newline='') as file:
    dict_of_sources = dict(iter(GlobalCases.groupby('Country/Region')))
    sumMSE=0
    for x, y in dict_of_sources.items():
        ActualData= y['Value'].values.astype(float)
        Lat= y['Lat'].values.astype(float)
        Long= y['Long'].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=(2,2,1), enforce_stationarity=False, enforce_invertibility=False)
            model_fit = model.fit()
            prediction = model_fit.forecast(180)
            #plt.figure(figsize=(10,5))
            #plt.plot(TestData,prediction, color='red')
            #MSEArima= (sum(prediction[0]-TestData)**2)/len(TestData)
            #sumMSE= sumMSE+MSEArima
            print(x,prediction)
            #print(MSEArima,x)
            writer = csv.writer(file,delimiter=',')
            writer.writerow([x,Lat[0],Long[0],prediction[13]])
```

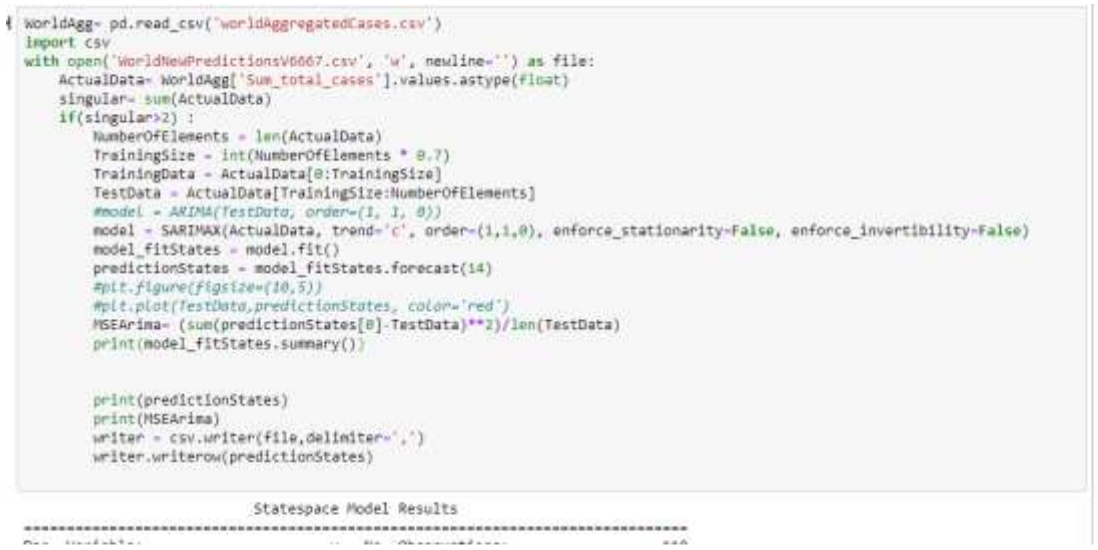
```
Afghanistan [  706.29634799   755.26036391   802.86896883   851.45899532
   900.59570942   950.31860273  1000.63807422  1051.54718179
  1103.04824844  1155.14071452  1207.82467395  1261.10012166
  1314.96705391  1369.42547262  1424.4753772   1480.11676779
  1536.34964437  1593.17400693  1650.58985548  1708.59719001
  1767.19601053  1826.38631704  1886.16810954  1946.54138802
  2007.50615249  2069.06240294  2131.21013938  2193.94936181
  2257.28007023  2321.20226463  2385.71594502  2450.8211114
  2516.51776376  2582.80590211  2649.68552644  2717.15663677
  2785.21923308  2853.87331537  2923.11888366  2992.95593792
  3063.38447818  3134.40450442  3206.01601665  3278.21901487
```

These predictions are useful for looking at future cases for a specific location or country, but they don't provide a clear picture of the rate of increase in total cases across the US or the world. I used Alteryx to aggregate the US and world data to give total US, and world cases by date.



date	Sum_total_cases	Sum_new_cases	Sum_new_deaths
########	27	27	0
1/1/2020	27	0	0
1/2/2020	27	0	0
1/3/2020	44	17	0
1/4/2020	44	0	0
1/5/2020	59	15	0
1/6/2020	59	0	0
1/7/2020	59	0	0
1/8/2020	59	0	0
1/9/2020	59	0	0
1/10/2020	59	0	0
1/11/2020	59	0	1
1/12/2020	59	0	0
1/13/2020	60	1	0
1/14/2020	60	0	0
1/15/2020	61	1	1
1/16/2020	61	0	0
1/17/2020	66	5	0
1/18/2020	83	17	0
1/19/2020	219	136	1
1/20/2020	239	20	0
1/21/2020	392	153	3
1/22/2020	534	142	11
1/23/2020	631	97	0
1/24/2020	897	266	9
1/25/2020	1350	453	15
1/26/2020	2023	673	15
1/27/2020	2820	797	25
1/28/2020	4587	1767	25
1/29/2020	6067	1480	26
1/30/2020	7823	1756	38

This data was fed into time series models using the parameters selected earlier, to predict the total number of cases.

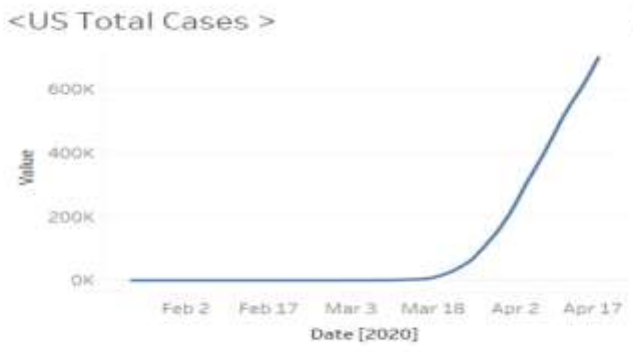


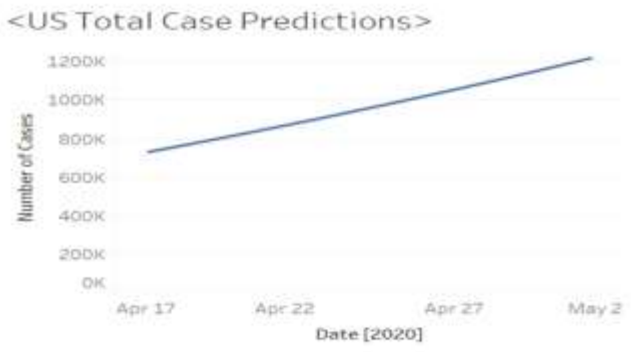
```
WorldAgg= pd.read_csv('worldAggregatedCases.csv')
import csv
with open('WorldNewPredictionsV6667.csv', 'w', newline='') as file:
    ActualData= WorldAgg['Sum_total_cases'].values.astype(float)
    singular= sum(ActualData)
    if(singular>2) :
        NumberOfElements = len(ActualData)
        TrainingSize = 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=(1,1,0), enforce_stationarity=False, enforce_invertibility=False)
        model_fitStates = model.fit()
        predictionStates = model_fitStates.forecast(14)
        #plt.figure(figsize=(10,5))
        #plt.plot(TestData,predictionStates, color='red')
        MSEArima= (sum(predictionStates[0]-TestData)**2)/len(TestData)
        print(model_fitStates.summary())


        print(predictionStates)
        print(MSEArima)
        writer = csv.writer(file,delimiter=',')
        writer.writerow(predictionStates)

                           Statespace Model Results
==============================================================================
```

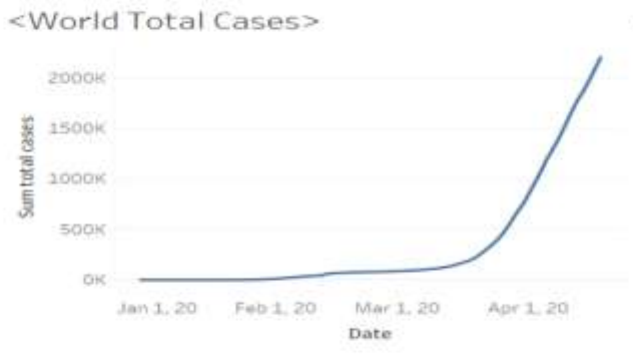
Total world predictions :

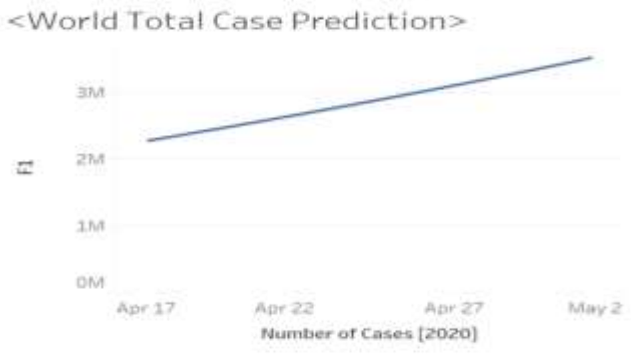
```
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 6.82e+22. Standard er
[2281657.58056652 2367869.72763426 2455548.58346622 2544709.44828452
 2635367.78191816 2727539.20546793 2821239.50298885 2916484.62318995
 3013290.68115196 3111673.96006281 3211650.91297127 3313238.1645589
 3416452.51293044 3521310.93142294]
```

I used these predictions to create charts to show the trend of total cases:









These charts indicate that the number of total cases will continue to increase in the US and the world.

Results

Using these time series model, I was unable to predict when the total number of cases will start to peak. All the predictions indicate an increase in total cases among most locations and countries with no slowdown, even when extending the forecast 3 to 6 months into the future. Of course no one knows exactly when the peak will occur, but realistically it is expected that the number of total cases will have leveled off 6 months from now. The models are unable to see the peak likely because they were trained using data from before the peak. A possible solution to this is to retrain the models using data from China, where the peak has already occurred (assuming there isn't a second wave). The problem is that case data from China has been somewhat unreliable, the data from Johns Hopkins indicated many locations in China were at or past the peak in total number of cases, but the total reported number of cases in the dataset is far lower than the actual number of cases. It appears China was initially under reporting cases which makes any models trained from that data unreliable when trying to predict the actual total number of cases for other locations/countries.

Kyle W. Brown

The COVID-19 Final Project for DSE 6300 was selected based primarily on the timing of the pandemic and its impact to us all and proposed by me. This section follows a top down format and structured to indicate my work contributed to the overall Project Schedule, goals, deadlines, and success. Words highlighted in blue are hyperlinked to their respective model, markdown file, and Jupyter notebook located in the COVID-19 project GitHub repo.

DSE 6300 COVID-19 GitHub Repo

My initial contribution with creating a GitHub repo to track work and progress. The COVID19-Project repo was created on April 1st and collaborator requests were sent to the entire group including the professor. Notices were also sent to the group immediately via a text, email, and/or discussion board post.

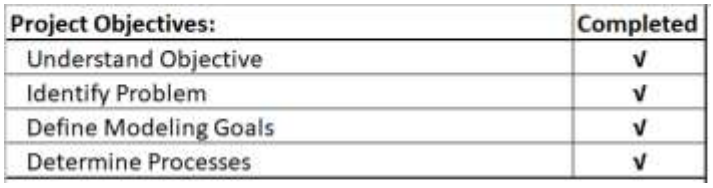
Project Proposal

Initially the Project Proposal was somewhat collaborated on from behalf of the team. Could not agree on any form of communication for several weeks. I addressed each question individually to the group then proceed to answer them myself which is my meaning of somewhat collaboration. After communicating roles with the team, I suggested: Derick: Pipeline, Jacqueline: Machine Learning, Ishaan: Project Manager, Kyle: Dashboard and Visualization, Nishchitha: Storage and Database.

Project Schedule

Laid out the Project Schedule by proposing to the group to follow a shortened agile framework focusing on 2 - (1) week iterations followed by, (1) week of either model integration and dashboard development, and (1) of dashboard deployment. With guidelines of the assignment we needed a Checkpoint 1 and Checkpoint 2 to indicate progress. The Checkpoints were identified by the Project Schedule and was adjusted to reflect the midterm Checkpoint at week (3). Updated the team prior to the first Checkpoint with the Initial Task list that lead to the development of the Agile Schedule. As advised by professor created a final week project schedule which was supposed to work back from the deadline to the day.

Screenshot of Initial Task List



Project Objectives:	Completed
Understand Objective	√
Identify Problem	√
Define Modeling Goals	√
Determine Processes	√

Final Week Project Schedule

Thu	Fri	Sat	Sun	Mon	Tue
4/16/2020	4/17/2020	4/18/2020	4/19/2020	4/20/2020	4/21/2020
Final Modeling	Model update, submit models by 7:00 p.m.	• Group model selection • Kafka setup	• Start ppt • Kafka topics, records, cluster	• Finish ppt parts •Finish pipeline	Finalize ppt

Checkpoint 1

1. What did you do this week?

- Setup GitHub repo.
- Combined several datasets from state level economic, environmental, and health factors. Also added a SARS dataset.
- Commit to repo with linear regression using R.
- Ran a time series analysis using Tensorflow and received an initial score of 18.5%.
- Using AutoML to explore potential deep learning, machine learning, and time series models then will export the model via an API.
    - Exploring different ways to train and test the model against SARS data.
- Standardized data from the initial dataset posted by John Hopkins University Github into 1 dataset based on Date, Province State, Country Region, Confirmed, Deaths, Recovered.
- Developed Task List.
- Attempting to configure SparkR, RStudio, and the Grid with the HPC team.

2. What are you planning to do next week?

- Finalize datasets and data sources.

- Make a commit to GitHub to complete linear regression from Regression_R file in GitHub.
- Complete a time series model expressing a metric like Mean Absolute Error using AutoML as well as the increase the Tensorflow notebook up from 18.5%
- Configure SparkR, shinyjs, Shiny, and the Grid correctly.
- Explore dashboard layouts and style.
- Prepare the data and setup Spark Stream using the Grid.

2. **What problems are you having?**
   - Initially, my biggest problem was overall project organization and understanding what tools could be used.
   - Lack of complete data for factors at the city/county/zip code level.
   - Cleansing the data for a standardized dataset.
   - X11 Forwarding gave me issues for display. Had to export dbus, first.
   - Time needed for model training, development, and selection.
   - SparkR was not an installed package on my node through the Grid.
   - Issues launching RStudio because of X11 Forwarding errors.

Checkpoint 2

1. **What did you do this week?**
   - Configured and installed the remaining components on my Grid RStudio notebook & server to use SparkR, sparklr, RStudio, Shiny, and Spark properly.
   - Completed linear, non-linear, and started a possible exponential regression with varying results.
   - Completed AutoML GBM Multinomial model that provided exceptional results.
   - Created a baseline time series with minor results.
   - Configured dashboard UI, global, server, and CSS folders in dashboard.
   - Made 5 commits this week to the GitHub.

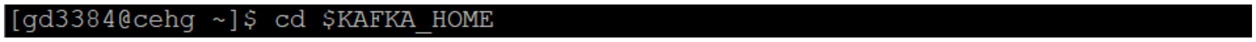
2. **What are you planning to do next week?**
   - Combine all models for presenting purposes.
   - Complete PowerPoint for final presentation.
   - Possibly adjust models for more complexity in operational dashboard.
   - Finalize dashboard CSS fonts, colors and style.
   - Complete final dashboard development.
   - Make final commit to COVID19 GitHub repo with final project accompanied with code, project schedule, data, and findings.

3. **What problems are you having?**
   - Difficulty communicating with group members.
   - No one follows up with updates or deadlines.

Pipeline

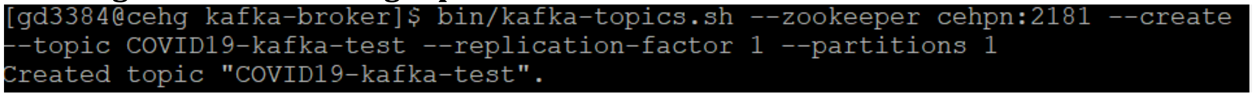
The goal of the original pipeline was to use Kafka and stream to the Grid using SparkR and Spark. Attention was given since the project objective centered around building a functional dashboard in Shiny presenting COVID19 virus predictions and streaming COVID data using Kafka or Spark Streaming. Unfortunately, time constraints played the biggest factor in the pipeline development for me.

Kafka directory to the Grid:

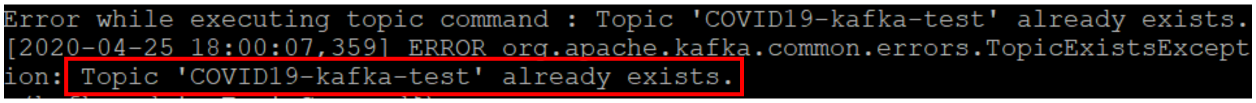


```
[gd3384@cehg ~]$ cd $KAFKA_HOME
```

Creating a COVID19 testing topic:

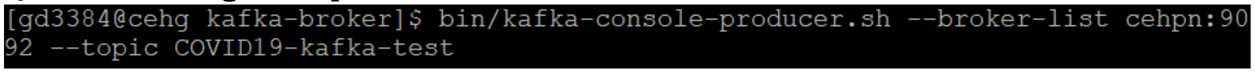


```
[gd3384@cehg kafka-broker]$ bin/kafka-topics.sh --zookeeper cehpn:2181 --create
--topic COVID19-kafka-test --replication-factor 1 --partitions 1
Created topic "COVID19-kafka-test".
```



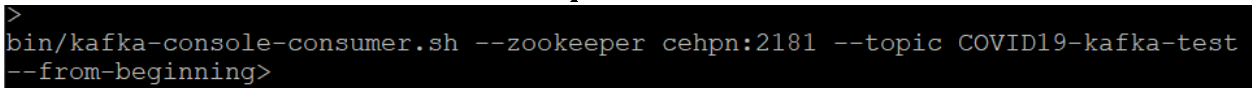
```
Error while executing topic command : Topic 'COVID19-kafka-test' already exists.
[2020-04-25 18:00:07,359] ERROR org.apache.kafka.common.errors.TopicExistsExcept
ion: Topic 'COVID19-kafka-test' already exists.
```

Quick test using the topic:



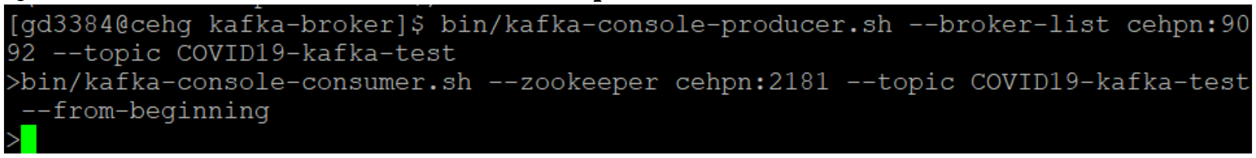
```
[gd3384@cehg kafka-broker]$ bin/kafka-console-producer.sh --broker-list cehpn:90
92 --topic COVID19-kafka-test
```

Consumer contents for the COVID19 topic:



```
>
bin/kafka-console-consumer.sh --zookeeper cehpn:2181 --topic COVID19-kafka-test
--from-beginning>
```

Quick test and Consumer contents for topic in command:

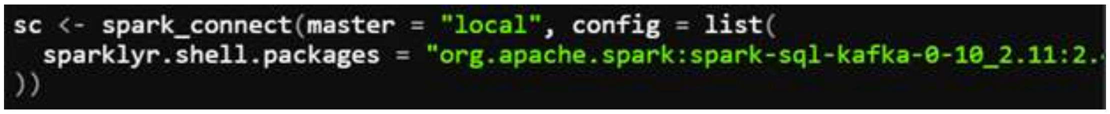


```
[gd3384@cehg kafka-broker]$ bin/kafka-console-producer.sh --broker-list cehpn:90
92 --topic COVID19-kafka-test
>bin/kafka-console-consumer.sh --zookeeper cehpn:2181 --topic COVID19-kafka-test
 --from-beginning
>
```

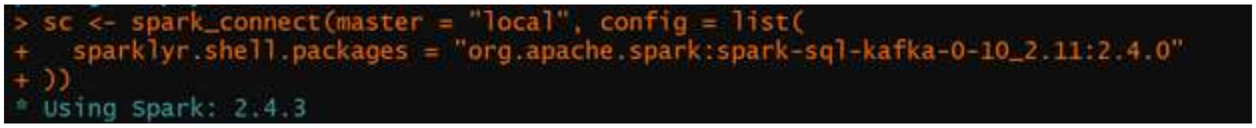
Connected SparkR and Kafka on the Grid

Connected Grid with SparkR and Kafka using WSU Ondemand HPC RStudio Server. Installed Spark into my RStudio environment to use SparkR on the Grid and Kafka for streaming but came up short of integration due to timing.

Connected SparkR and Kafka to the Grid

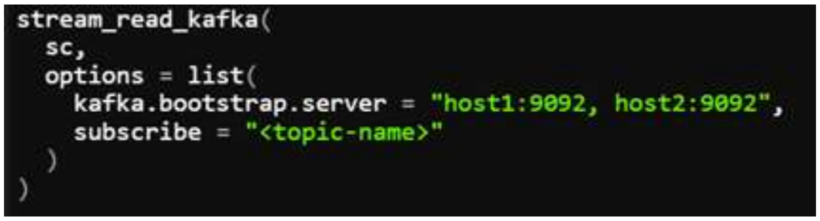


```
sc <- spark_connect(master = "local", config = list(
  sparklyr.shell.packages = "org.apache.spark:spark-sql-kafka-0-10_2.11:2.
))
```



```
> sc <- spark_connect(master = "local", config = list(
+   sparklyr.shell.packages = "org.apache.spark:spark-sql-kafka-0-10_2.11:2.4.0"
+ ))
* Using Spark: 2.4.3
```

Example setup to read Kafka stream.



```
stream_read_kafka(
  sc,
  options = list(
    kafka.bootstrap.server = "host1:9092, host2:9092",
    subscribe = "<topic-name>"
  )
)
```

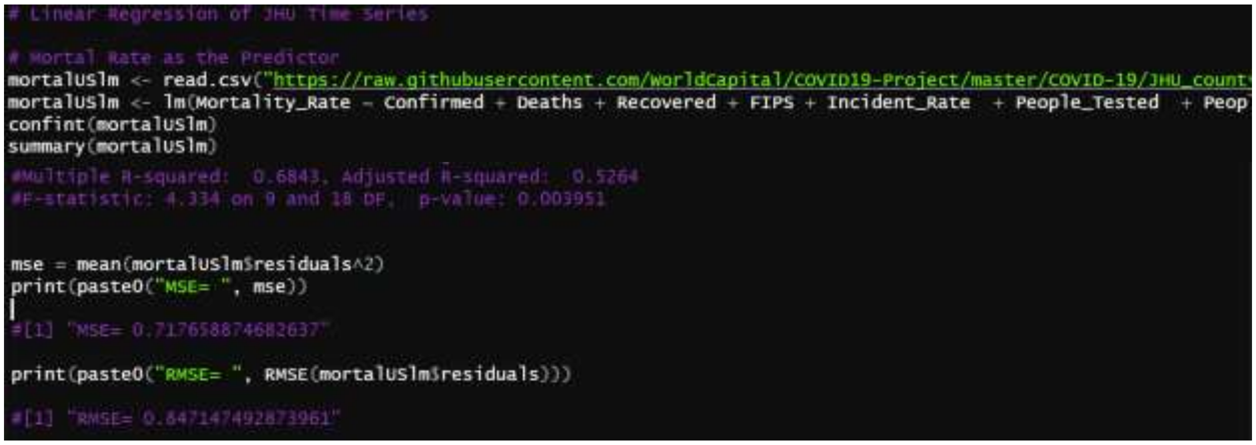
Models

The modeling process began with a linear regression and progressed into AutoML GBM Multinomial model. The initial Linear Regression was posted on 4/4/2020 as Regression_R to the project GitHub with 20+ models compiled using CDC data. The remaining models were created from either STATWORX COVID19 API and John Hopkins University GitHub (both files are included in repo). On 4/11/2020, a Models Update R Markdown file containing several linear, non-linear, and principal component analysis models were added to the repo.

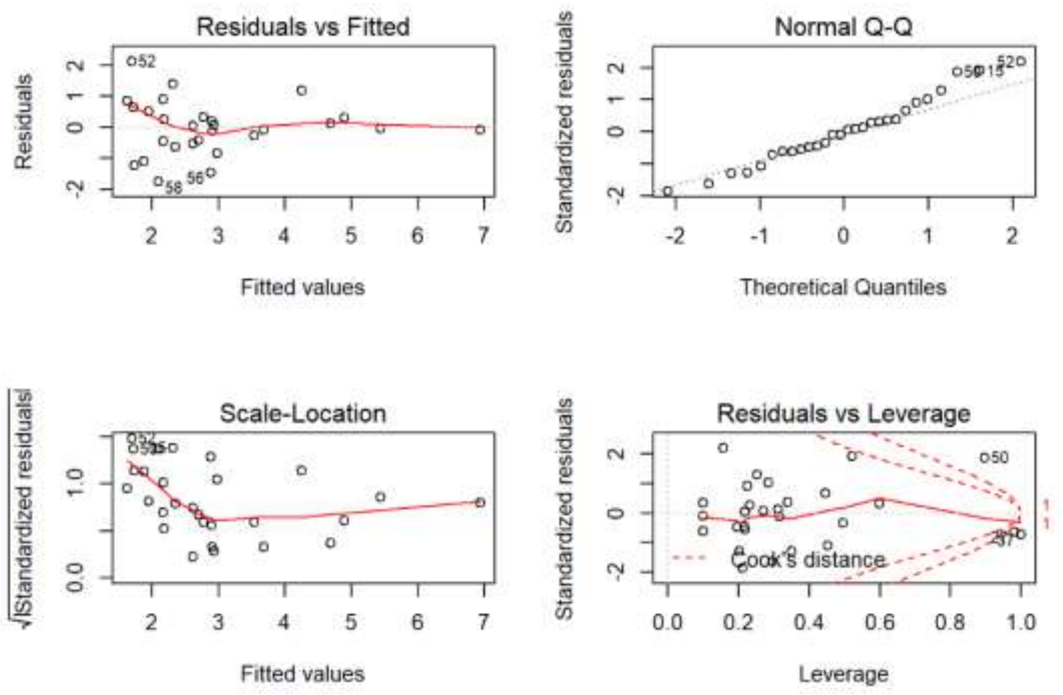
The models used were:

- Linear Regression
- Multivariate Regression
- Non-Linear Regression
- Principal Component Analysis
- Tensorflow Time Series
- AutoML Deep Learning

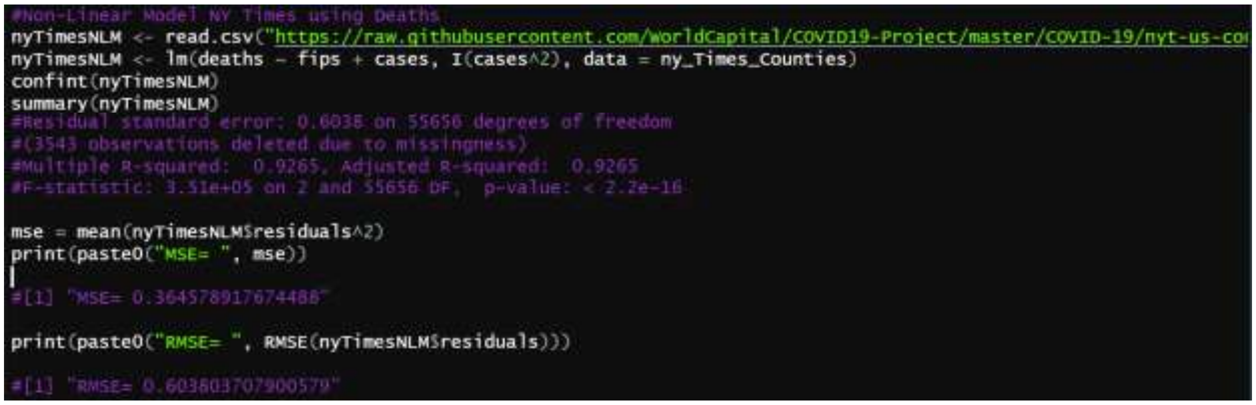
Linear Regressions



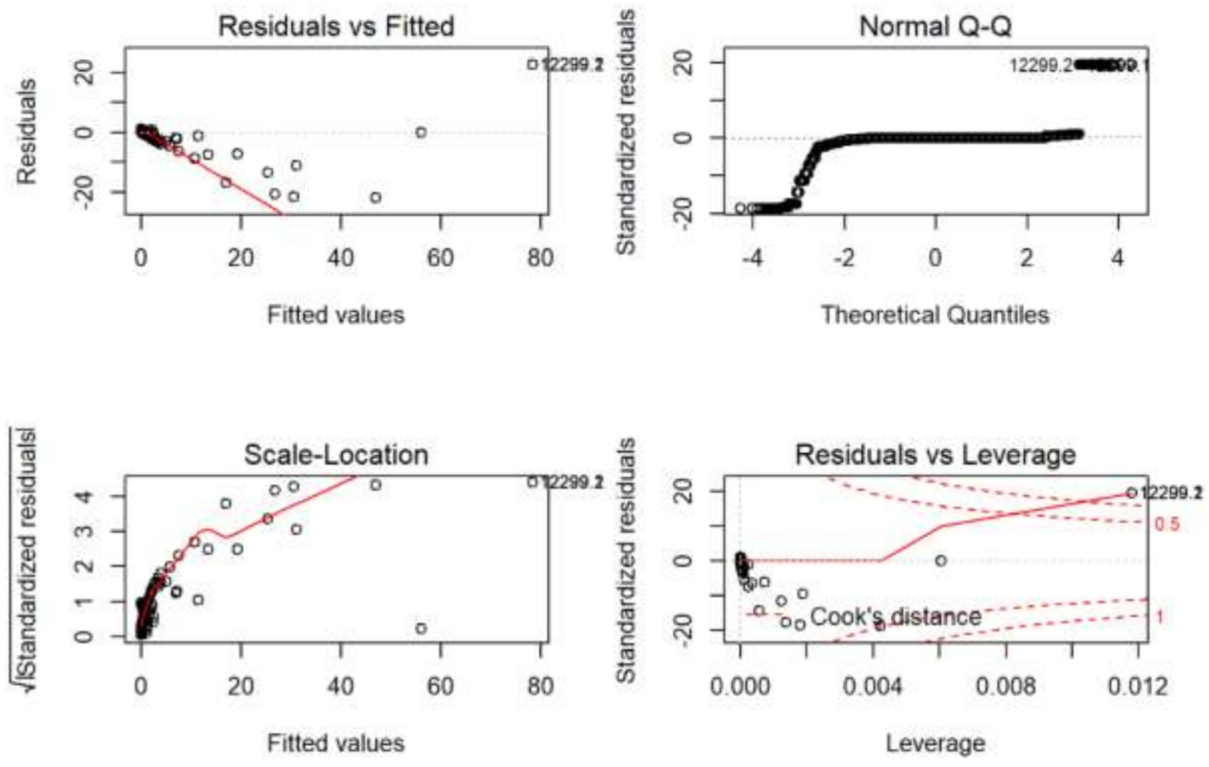
```
# Linear Regression of JHU Time Series

# Mortal Rate as the Predictor
mortalUSlm <- read.csv("https://raw.githubusercontent.com/WorldCapital/COVID19-Project/master/COVID-19/JHU_count
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"
```

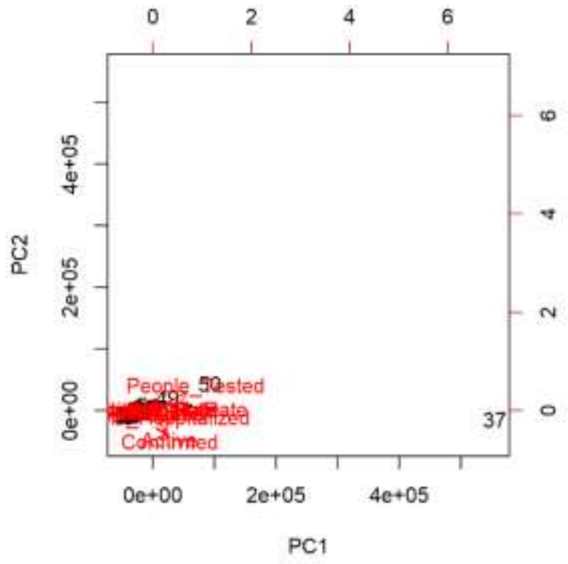


Principal Component Analysis (PCA)

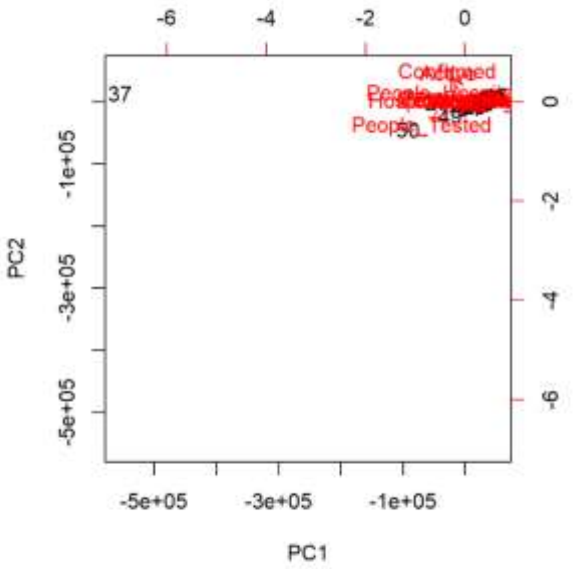
```
# Principal Component Analysis

countUS <- subset(countyUS, select = -c(Province_State, Last_Update,Country_Region,UID, Lat, Long_, ISO3))

countUS <- na.omit(countUS)

countUS <- transform(countUS, Confirmed = as.numeric(Confirmed),
          Deaths = as.numeric(Deaths),
          Recovered = as.numeric(Recovered),
          Active = as.numeric(Active),
          FIPS = as.numeric(FIPS),
          People_Tested = as.numeric(People_Tested),
          People_Hospitalized = as.numeric(People_Hospitalized))

apply(countUS, 2, mean)
```

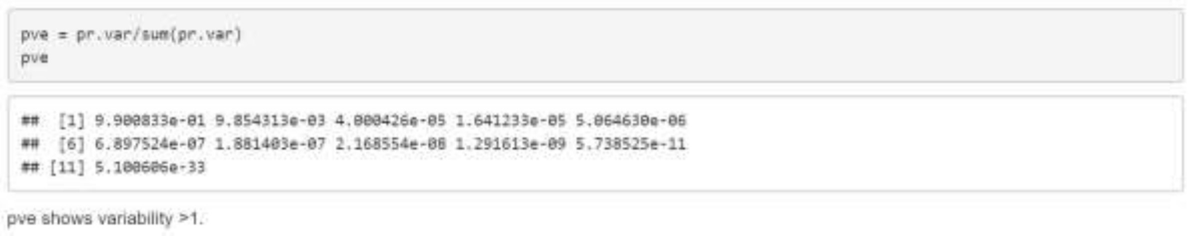
```
##          Confirmed              Deaths           Recovered
##       11165.321429          553.250000         1405.750000
##             Active                FIPS       Incident_Rate
##       10612.071429           32.500000          150.369843
##      People_Tested People_Hospitalized      Mortality_Rate
##       49492.428571         2158.714286            2.949897
##       Testing_Rate Hospitalization_Rate
##        1274.784932           14.246080
```

```
biplot(pr.out, scale =0)
```



```
pr.out$rotation = -pr.out$rotation
pr.out$x = -pr.out$x
biplot(pr.out, scale =0)
```





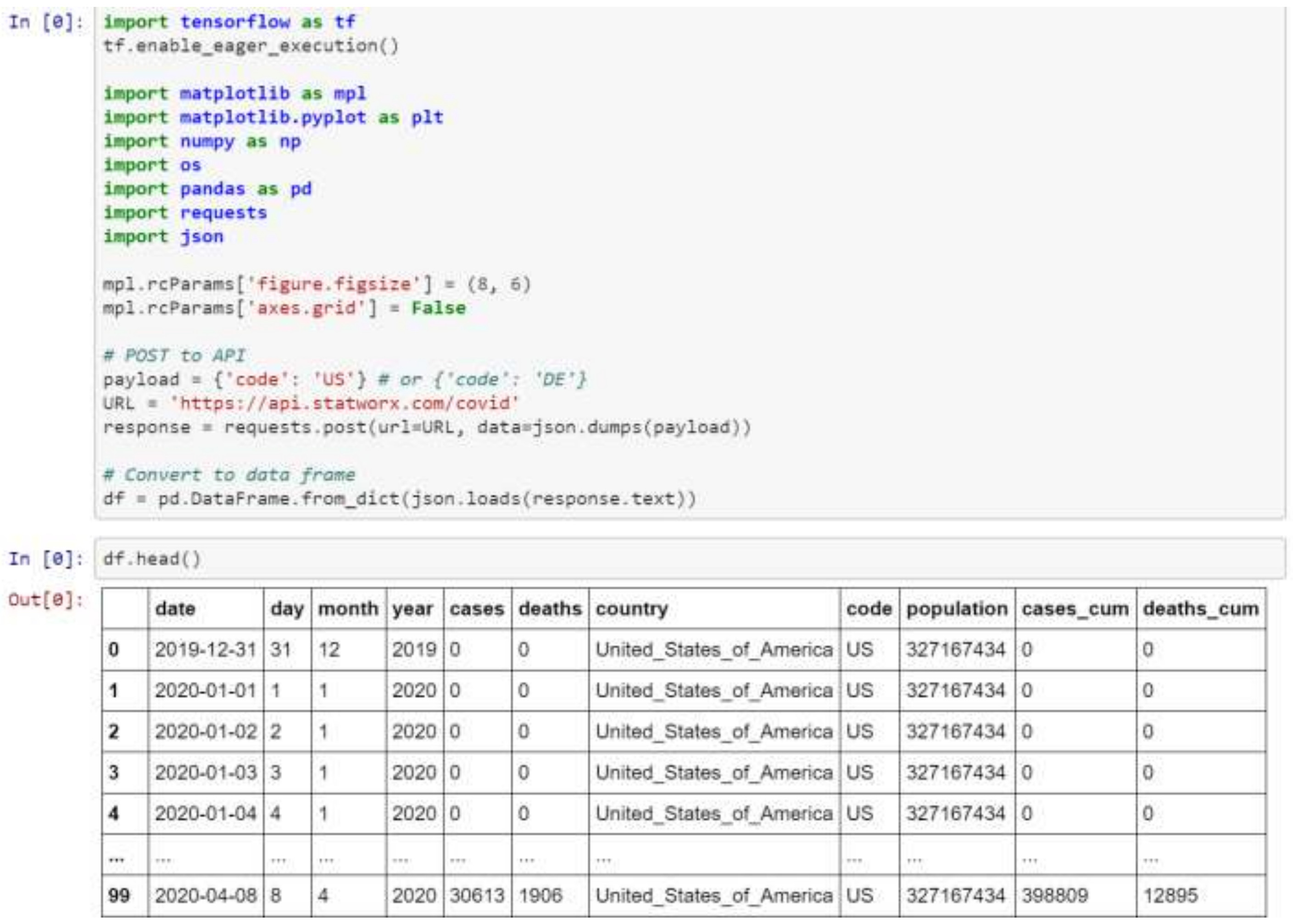
```
pve = pr.var/sum(pr.var)
pve
```

```
##  [1] 9.900833e-01 9.854313e-03 4.000426e-05 1.641233e-05 5.064630e-06
##  [6] 6.897524e-07 1.881403e-07 2.168554e-08 1.291613e-09 5.738525e-11
## [11] 5.100606e-33
```

pve shows variability >1.

Tensorflow Time Series

The Tensorflow Time Series performed



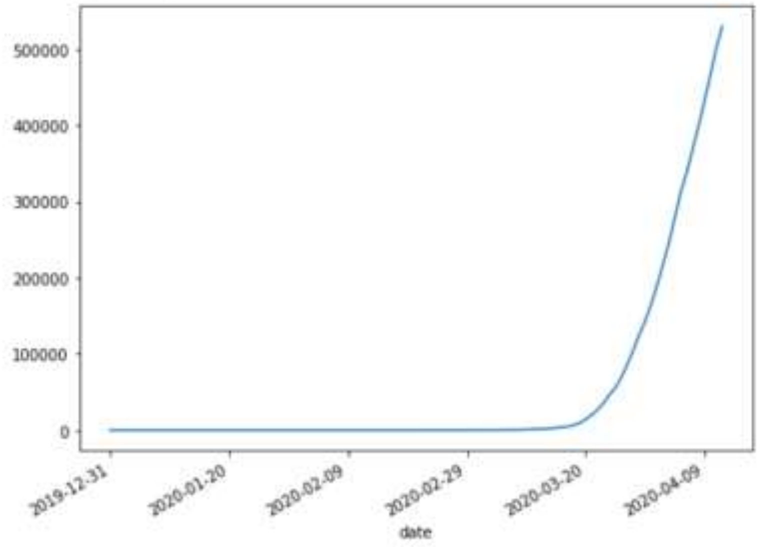
```
In [0]: import tensorflow as tf
        tf.enable_eager_execution()

        import matplotlib as mpl
        import matplotlib.pyplot as plt
        import numpy as np
        import os
        import pandas as pd
        import requests
        import json

        mpl.rcParams['figure.figsize'] = (8, 6)
        mpl.rcParams['axes.grid'] = False

        # POST to API
        payload = {'code': 'US'} # or {'code': 'DE'}
        URL = 'https://api.statworx.com/covid'
        response = requests.post(url=URL, data=json.dumps(payload))

        # Convert to data frame
        df = pd.DataFrame.from_dict(json.loads(response.text))
```

```
In [0]: df.head()
```

Out[0]:

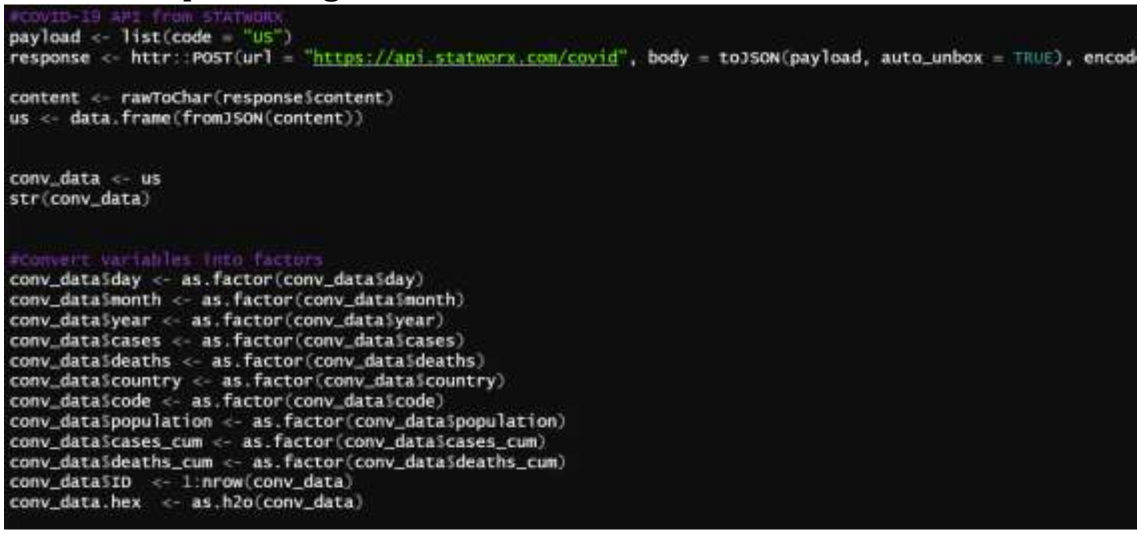
	date	day	month	year	cases	deaths	country	code	population	cases_cum	deaths_cum
0	2019-12-31	31	12	2019	0	0	United_States_of_America	US	327167434	0	0
1	2020-01-01	1	1	2020	0	0	United_States_of_America	US	327167434	0	0
2	2020-01-02	2	1	2020	0	0	United_States_of_America	US	327167434	0	0
3	2020-01-03	3	1	2020	0	0	United_States_of_America	US	327167434	0	0
4	2020-01-04	4	1	2020	0	0	United_States_of_America	US	327167434	0	0
...	...	...	...	...	...	...	...	...	...	...	...
99	2020-04-08	8	4	2020	30613	1906	United_States_of_America	US	327167434	398809	12895

```
In [0]: uni_data.plot(subplots=True)

Out[0]: array([<matplotlib.axes._subplots.AxesSubplot object at 0x7ff79e3c4cf8>],
              dtype=object)
```

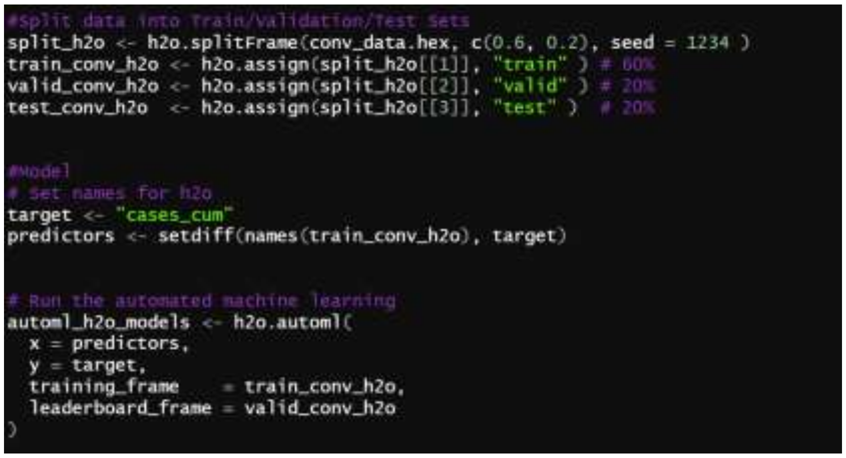


```
In [0]: print ('Single window of past history')
        print (x_train_uni[0])
        print ('\n Target cumulative confirmed to predict')
        print (y_train_uni[0])

        Single window of past history
        [[-0.35604638]
         [-0.35604638]
         [-0.35604638]
         [-0.35604638]
         [-0.35604638]
         [-0.35604638]
         [-0.35604638]
         [-0.35604638]
         [-0.35604638]
         [-0.35604638]
         [-0.35604638]
         [-0.35604638]
         [-0.35604638]
         [-0.35604638]
         [-0.35604638]
         [-0.35604638]
         [-0.35604638]
         [-0.35604638]
         [-0.35604638]
         [-0.35604638]]

         Target cumulative confirmed to predict
        -0.3560463830477833
```

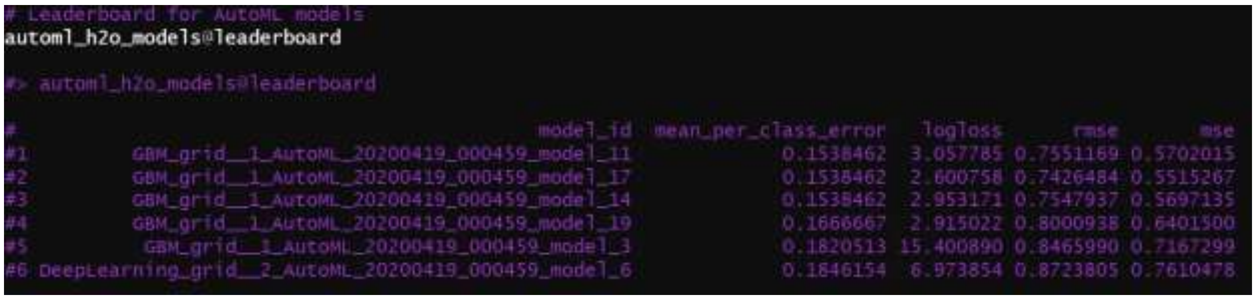
AutoML Deep Learning Models



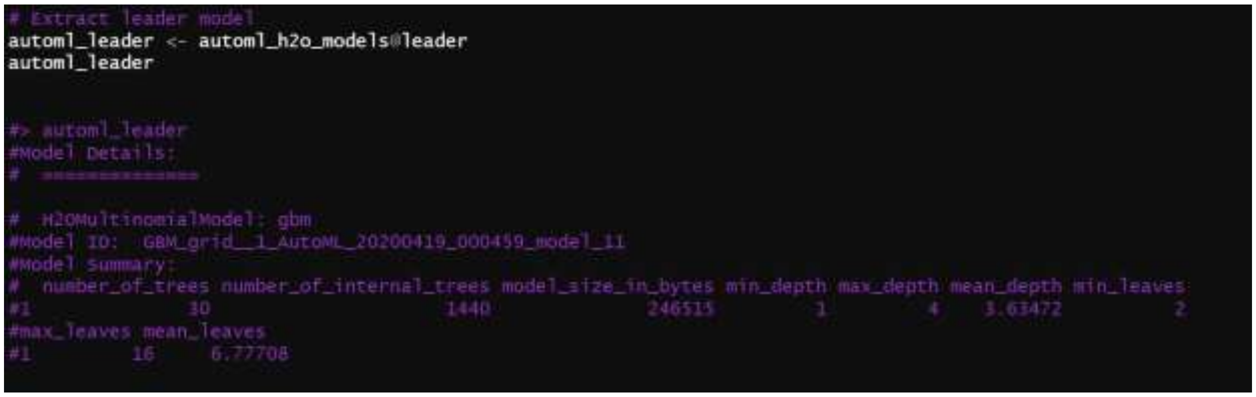
```r
#COVID-19 API from STATWORX
payload <- list(code = "US")
response <- httr::POST(url = "https://api.statworx.com/covid", body = toJSON(payload, auto_unbox = TRUE), encod

content <- rawToChar(response$content)
us <- data.frame(fromJSON(content))


conv_data <- us
str(conv_data)


#Convert variables into factors
conv_data$day <- as.factor(conv_data$day)
conv_data$month <- as.factor(conv_data$month)
conv_data$year <- as.factor(conv_data$year)
conv_data$cases <- as.factor(conv_data$cases)
conv_data$deaths <- as.factor(conv_data$deaths)
conv_data$country <- as.factor(conv_data$country)
conv_data$code <- as.factor(conv_data$code)
conv_data$population <- as.factor(conv_data$population)
conv_data$cases_cum <- as.factor(conv_data$cases_cum)
conv_data$deaths_cum <- as.factor(conv_data$deaths_cum)
conv_data$ID  <- 1:nrow(conv_data)
conv_data.hex  <- as.h2o(conv_data)
```



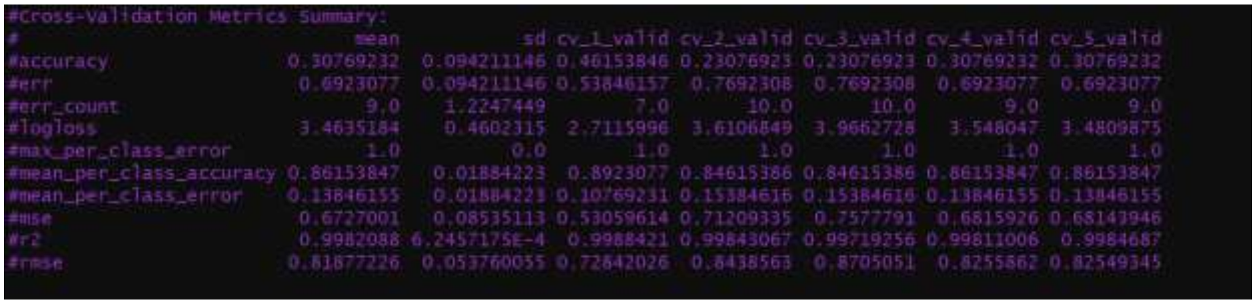
```r
#Split data into Train/Validation/Test Sets
split_h2o <- h2o.splitFrame(conv_data.hex, c(0.6, 0.2), seed = 1234 )
train_conv_h2o <- h2o.assign(split_h2o[[1]], "train" ) # 60%
valid_conv_h2o <- h2o.assign(split_h2o[[2]], "valid" ) # 20%
test_conv_h2o  <- h2o.assign(split_h2o[[3]], "test" )  # 20%


#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
)
```



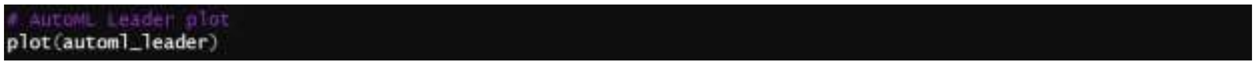
```r
# Leaderboard for AutoML models
automl_h2o_models@leaderboard

#> automl_h2o_models@leaderboard

#                                              model_id mean_per_class_error   logloss      rmse       mse
#1       GBM_grid__1_AutoML_20200419_000459_model_11            0.1538462  3.057785 0.7551169 0.5702015
#2       GBM_grid__1_AutoML_20200419_000459_model_17            0.1538462  2.600758 0.7426484 0.5515267
#3       GBM_grid__1_AutoML_20200419_000459_model_14            0.1538462  2.953171 0.7547937 0.5697135
#4       GBM_grid__1_AutoML_20200419_000459_model_19            0.1666667  2.915022 0.8000938 0.6401500
#5        GBM_grid__1_AutoML_20200419_000459_model_3            0.1820513 15.400890 0.8465990 0.7167299
#6 DeepLearning_grid__2_AutoML_20200419_000459_model_6            0.1846154  6.973854 0.8723805 0.7610478
```



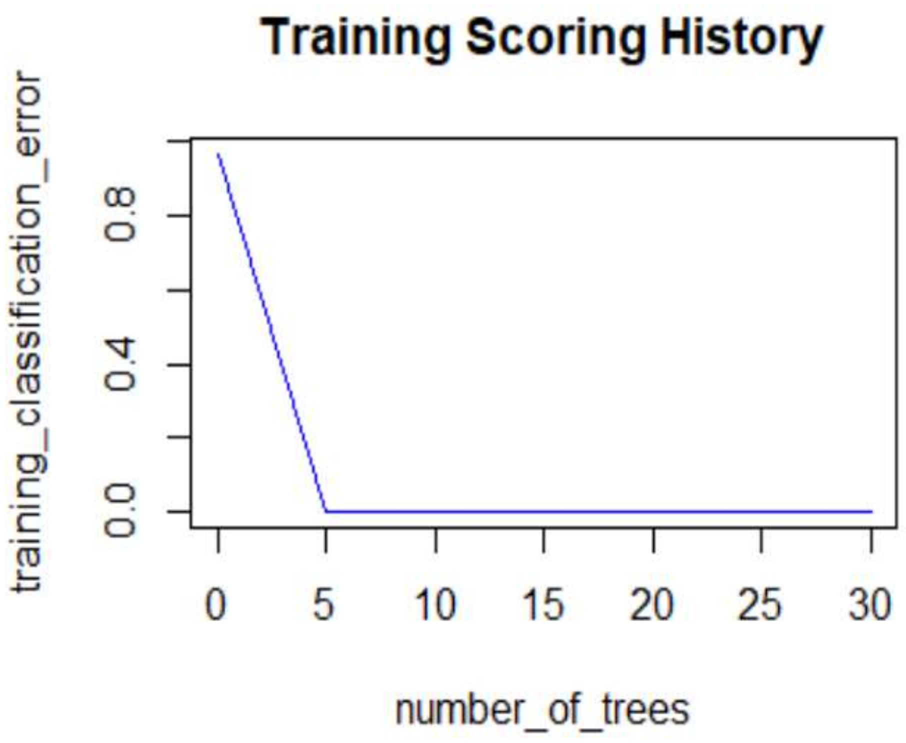
```
# Extract leader model
automl_leader <- automl_h2o_models@leader
automl_leader


#> automl_leader
#Model Details:
#  ==============

#  H2OMultinomialModel: gbm
#Model ID:  GBM_grid__1_AutoML_20200419_000459_model_11
#Model Summary:
#  number_of_trees number_of_internal_trees model_size_in_bytes min_depth max_depth mean_depth min_leaves
#1              30                     1440              246515         1         4    3.63472          2
#max_leaves mean_leaves
#1         16     6.77708
```



```
#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
```



```
# AutoML Leader plot
plot(automl_leader)
```



Top Models 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%

Top Performer Based on AutoML Leader

	R^2	MSE	RMSE
cases_cum_gbm_model	99.81%	67.69%	82.16%

Models Summary

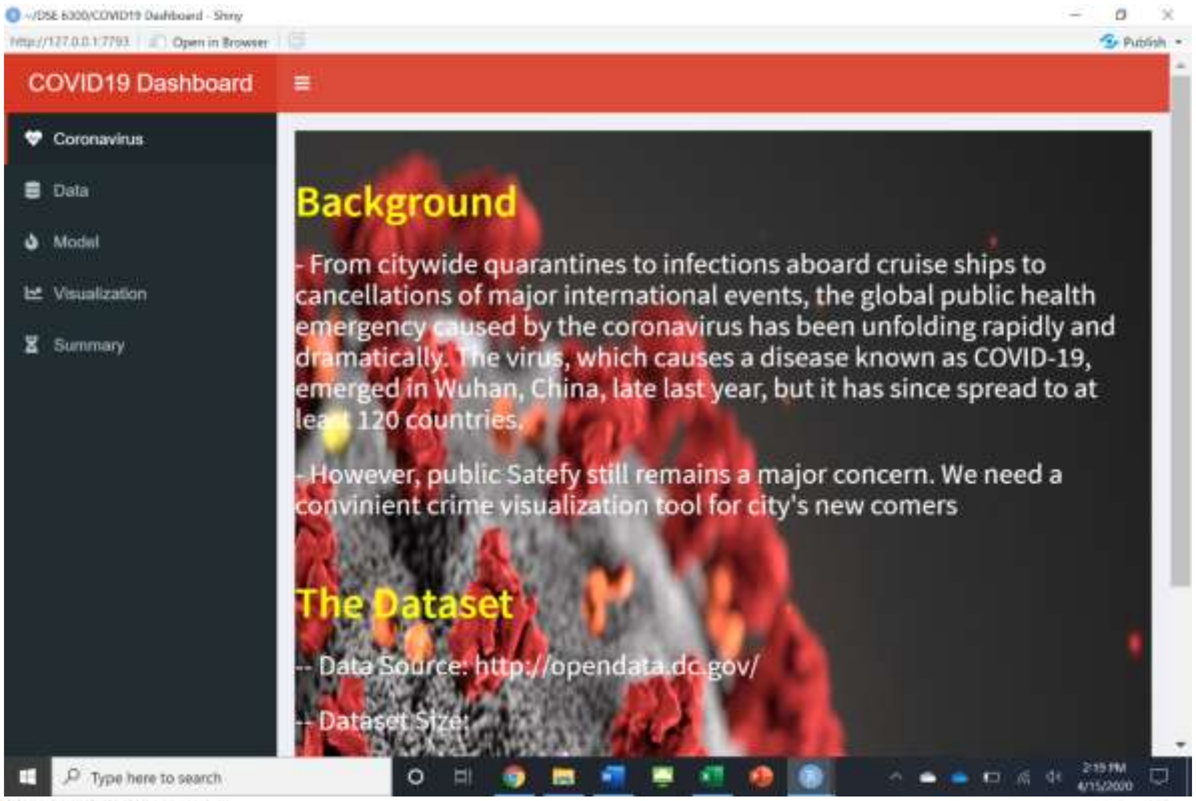
In summary, the modeling needed another week to deploy on real data. AutoML provided the best models with results ranging from 99% R-Squared, 74%-87% RMSE, and 55%-76% MSE. These models far outperform any of the R linear, non-linear, and exponential regressions, PCA, and Tensorflow Time Series. It is worth noting that Deaths JHU non-linear model outperformed Mortality Rate JHU linear model in R-Squared 92.65% to 52.64%. The Tensorflow Time Series only baselined 36.5% accuracy but not much better. The PCA PVE was greater than 1 rendering it useless.

Dashboards

Shiny Dashboard

The COVID19 Shiny Dashboard went into development once the topic for the project was selected. The COVID19 Dashboard directories, UI, server, CSS, and deployment was configured. The project ran out of time to incorporate the Grid and SparkR due to lack of pipeline and streams throughout the project. Attached below is a screenshot of the Shiny dashboard testing I used for initial testing using the STATWORX API. Once Shiny and SparkR became unfeasible, the project migrated to Tableau for dashboard development.

Shiny Dashboard Screenshots:



Background

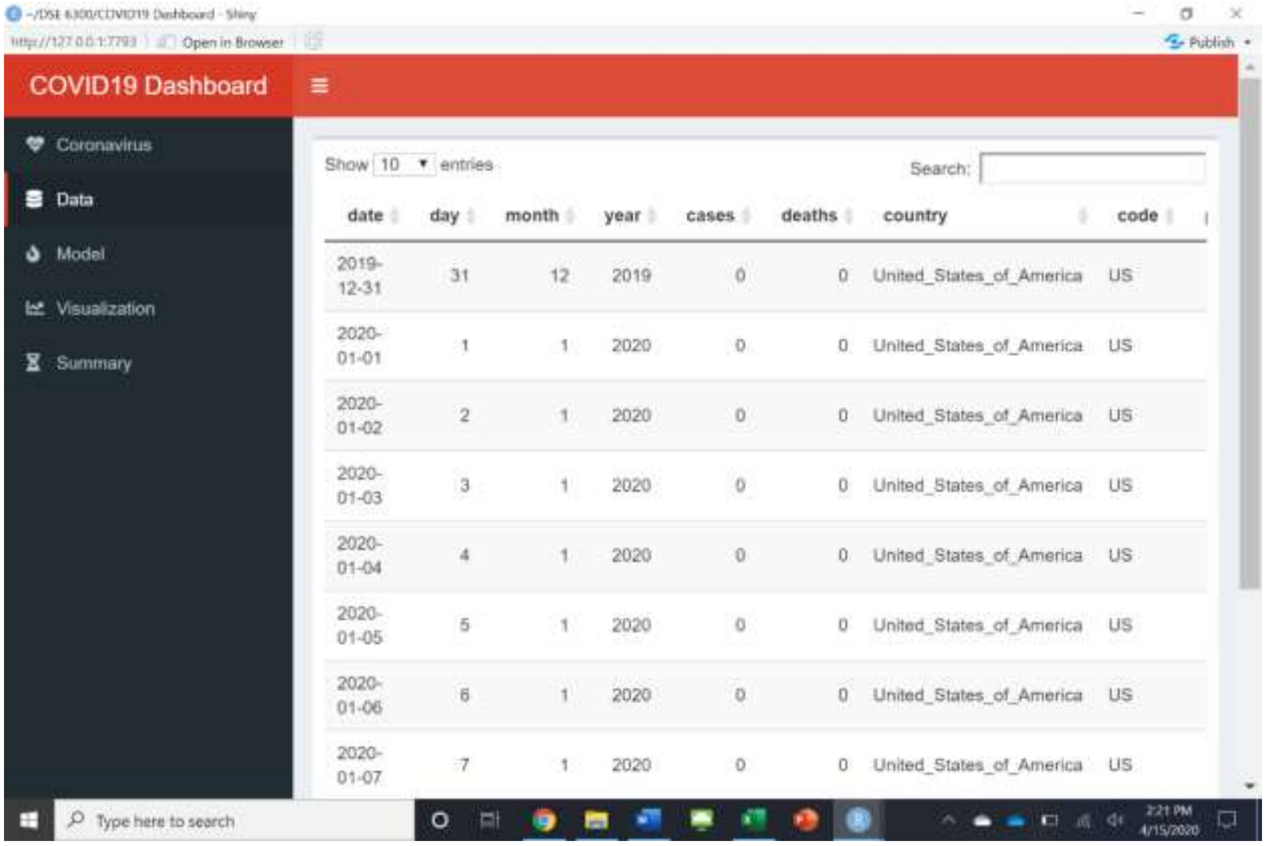
- From citywide quarantines to infections aboard cruise ships to cancellations of major international events, the global public health emergency caused by the coronavirus has been unfolding rapidly and dramatically. The virus, which causes a disease known as COVID-19, emerged in Wuhan, China, late last year, but it has since spread to at least 120 countries.

- However, public Satefy still remains a major concern. We need a convinient crime visualization tool for city's new comers

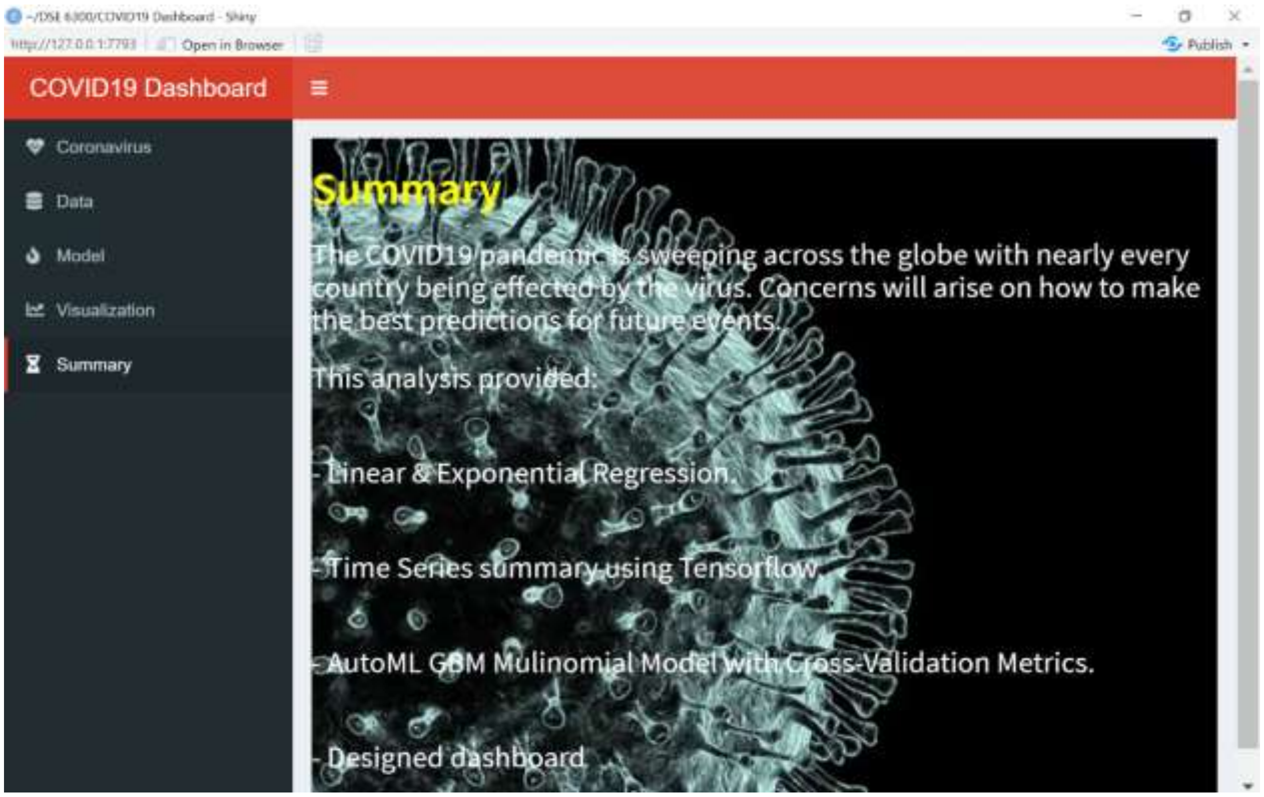
The Dataset

-- Data Source: http://opendata.dc.gov/

-- Dataset Size:



date	day	month	year	cases	deaths	country	code
2019-12-31	31	12	2019	0	0	United_States_of_America	US
2020-01-01	1	1	2020	0	0	United_States_of_America	US
2020-01-02	2	1	2020	0	0	United_States_of_America	US
2020-01-03	3	1	2020	0	0	United_States_of_America	US
2020-01-04	4	1	2020	0	0	United_States_of_America	US
2020-01-05	5	1	2020	0	0	United_States_of_America	US
2020-01-06	6	1	2020	0	0	United_States_of_America	US
2020-01-07	7	1	2020	0	0	United_States_of_America	US



Summary

The COVID19 pandemic is sweeping across the globe with nearly every country being effected by the virus. Concerns will arise on how to make the best predictions for future events.

This analysis provided:

- Linear & Exponential Regression.

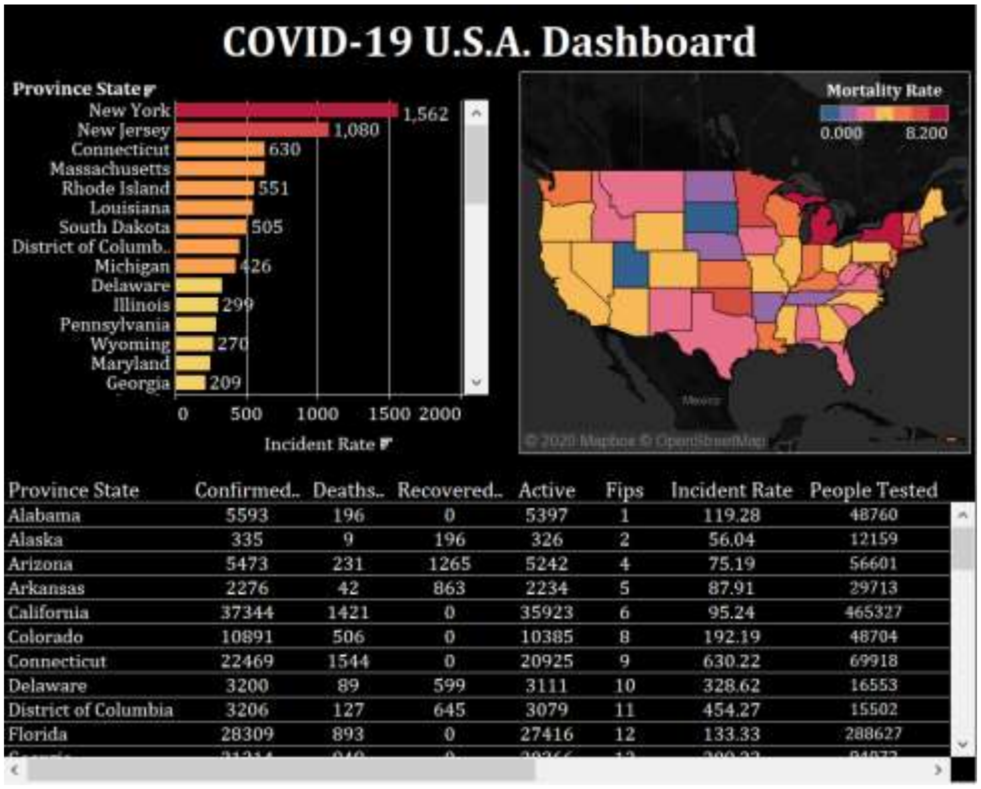
- Time Series summary using Tensorflow.

- AutoML GBM Mulinomial Model with Cross-Validation Metrics.

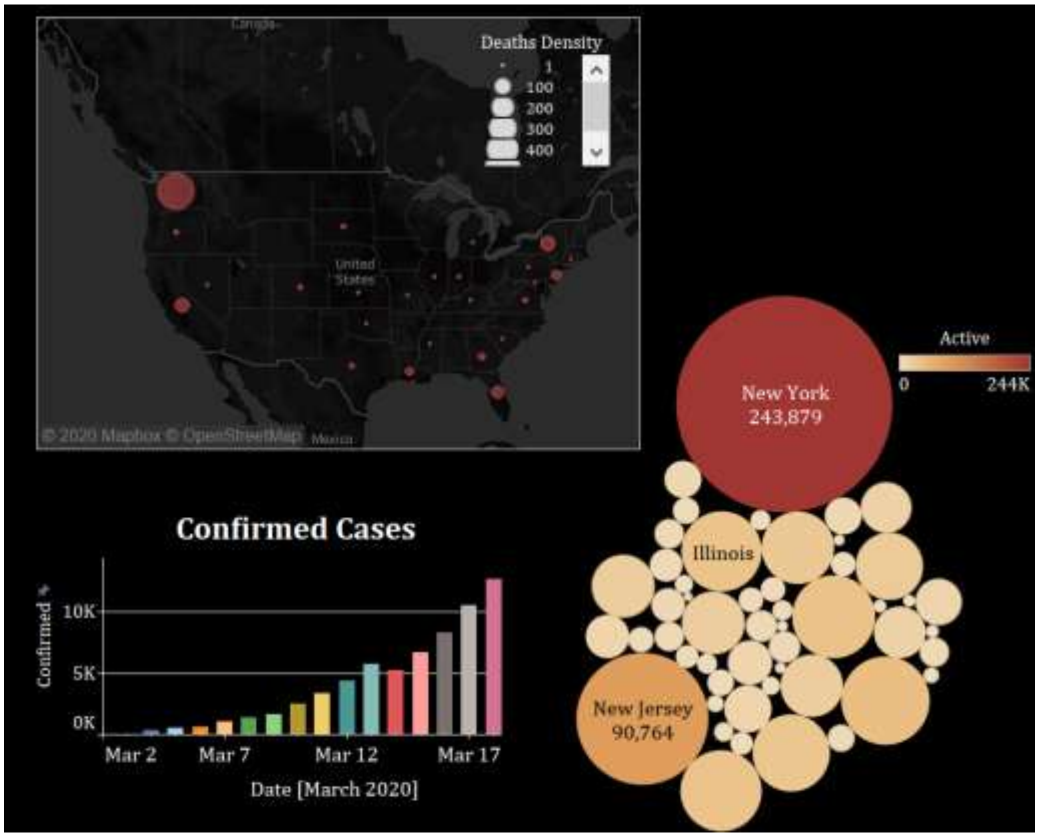
- Designed dashboard

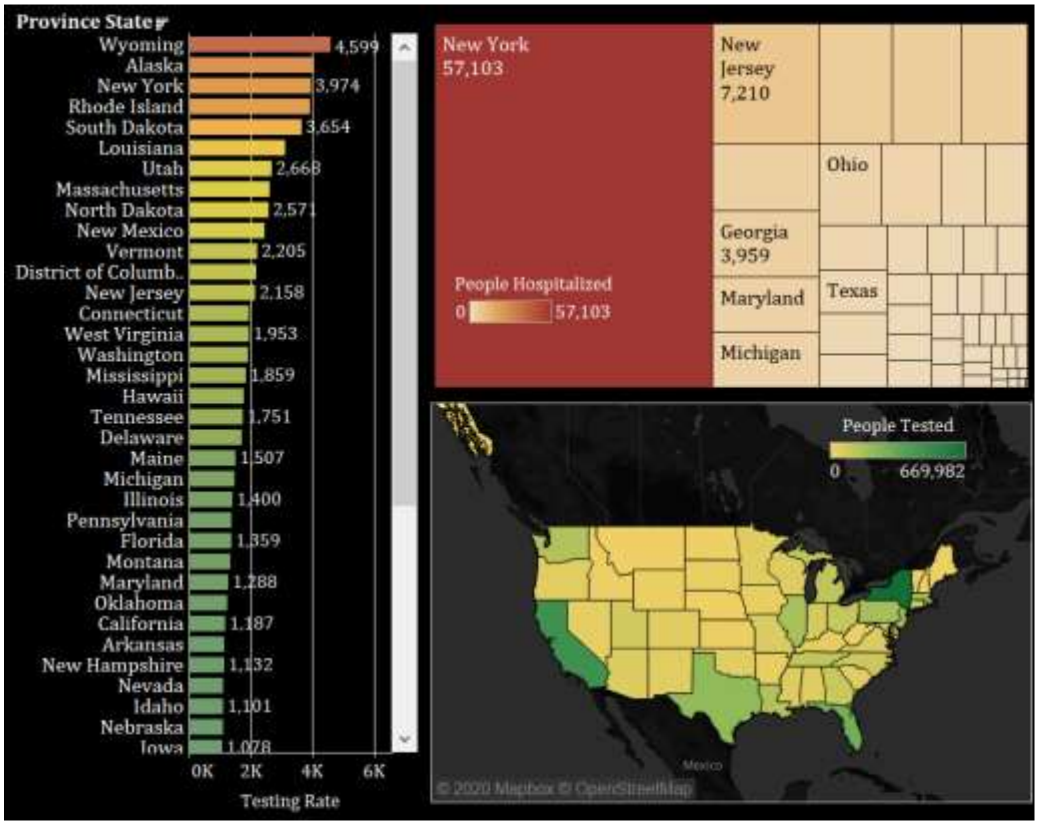
Tableau Dashboard

Tableau Dashboard Screenshots:



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
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]


Deaths Density
1
100
200
300
400
United States
© 2020 Mapbox © OpenStreetMap
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
Alaska
New York
Rhode Island
South Dakota
Louisiana
Utah
Massachusetts
North Dakota
New Mexico
Vermont
District of Columb..
New Jersey
Connecticut
West Virginia
Washington
Mississippi
Hawaii
Tennessee
Delaware
Maine
Michigan
Illinois
Pennsylvania
Florida
Montana
Maryland
Oklahoma
California
Arkansas
New Hampshire
Nevada
Idaho
Nebraska
Iowa
4,599
3,974
3,654
2,668
2,571
2,205
2,158
1,953
1,859
1,751
1,507
1,400
1,359
1,288
1,187
1,132
1,101
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
© 2020 Mapbox © OpenStreetMap

Summary

Still more conclusions needed to be drawn from the entire project. The GBM Multinomial model provided exceptional results based on time constraints with 99.81% R-Square, 67.69% MSE, and 82.16% RSME. Collaboration should have been the focus but due to external circumstance, communication time was limited (i.e. corona). The COVID19-Project repo will be updated with the final project and results. As an investor and customer, I would politely pass on funding this product/project.

Nishchitha Manjunath: Visualization

We from the DSE 6300 chose to collaborate and work on the COVID-19 data. Here, in this report I submit what my contributions were for the project with my checkpoints, schedule, tableau file, python file along with other deliverables.

Project Proposal

During the initial stages of our project , we were little organized in managing stuffs, but later due to the lockdown situation, which caused us to take remote classes, we missed several conversations and discussions regarding the project. It became less interactive with no updates on the project for several days. I wanted to do the visualization on tableau and so I chose to do the visualization for the project .

Project Schedule

There were two checkpoints 1 & 2 which were for used for tracking the accomplishments , plans for the checkpoint 2 and the hurdles on the way. As advised by professor we had a conference to talk about the project and on what needs to be done.

Checkpoint 1

1. What did I do this week?
   - Setup GitHub repository
   - Referred different datasets related to COVID-19
   - Configured my setup on google colab
   - Doing analysis of dataset with spark operations
   - Exploring the dataset for cleaning and visualization

2. What are you planning to do next week?
   - Configure my setup in a better way
   - Use the datasets and do analysis and cleaning on it
   - Figure out different plots
   - Use R to run my operations
   - Kafka streaming

3. What problems are you having?

- Getting to know which tool / language is used by whom and how do I work using it
- Cleansing the data for visualization.
- Language to consider for visualization
- Vast dataset and how accurate are we with regard to data.

Checkpoint 2

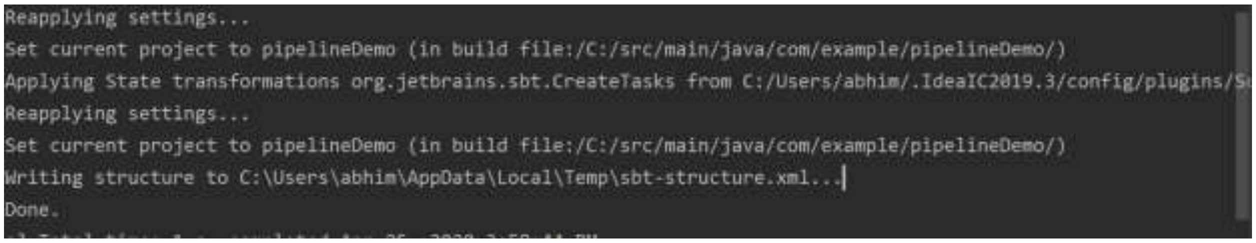
1. What did I do this week?
   - Setup GitHub repository.
   - Referred different datasets related to COVID-19
   - Configured my setup on tableau
   - Exploring the dataset for visualization

2. What are you planning to do next week?
   - Use the datasets and do analysis and be ready for visualization
   - Figure out different plots
   - Complete the Power Point Presentation for final submission
   - Plot many more graphs regard to the data
   - Complete the project well within the time frame.

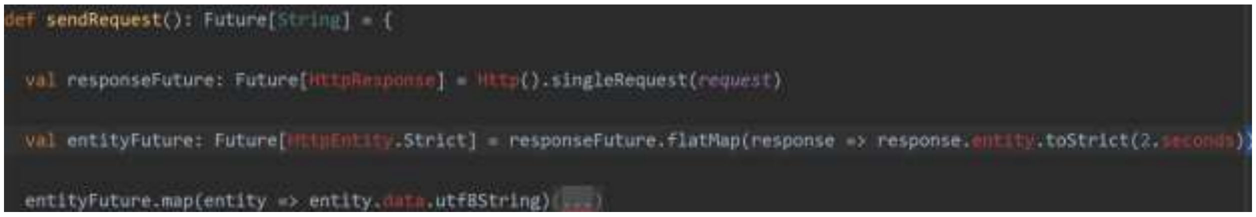
3. What problems are you having?
   - Few things are new in Tableau, so quite challenging the work, but will get it done
   - Getting the latest dataset downloaded, as GitHub RAW doesn't work accurately.

Pipeline

For Pipeline we had to initially use Kafka and stream it on the WSU grid using Scala and spark. Here is my trial for setting up the pipeline using IntelliJ using Scala framework. Below is an example where I tried to execute Jacqueline's code using IntelliJ but unfortunately got a lot of errors and didn't proceed much with it.



```
Reapplying settings...
Set current project to pipelineDemo (in build file:/C:/src/main/java/com/example/pipelineDemo/)
Applying State transformations org.jetbrains.sbt.CreateTasks from C:/Users/abhim/.IdeaIC2019.3/config/plugins/S
Reapplying settings...
Set current project to pipelineDemo (in build file:/C:/src/main/java/com/example/pipelineDemo/)
Writing structure to C:\Users\abhim\AppData\Local\Temp\sbt-structure.xml...
Done.
```



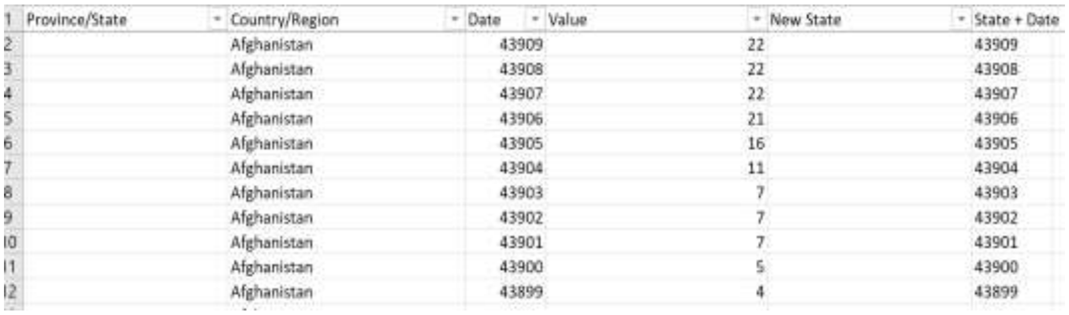
```
def sendRequest(): Future[String] = {

  val responseFuture: Future[HttpResponse] = Http().singleRequest(request)

  val entityFuture: Future[HttpEntity.Strict] = responseFuture.flatMap(response => response.entity.toStrict(2.seconds))

  entityFuture.map(entity => entity.data.utf8String)(...)
```

Tableau Visualizations

I chose the daily updated time series dataset from the John Hopkins University where I wanted to visualize the scenario of the COVID-19 pandemic just after the National Emergency in the US was declared. There were 3 datasets which had the details of Confirmed, Deaths and Recovered records, which were updated daily. I used Excel to merge the three datasets into one dataset and cleaned the data by doing some data scrubbing.

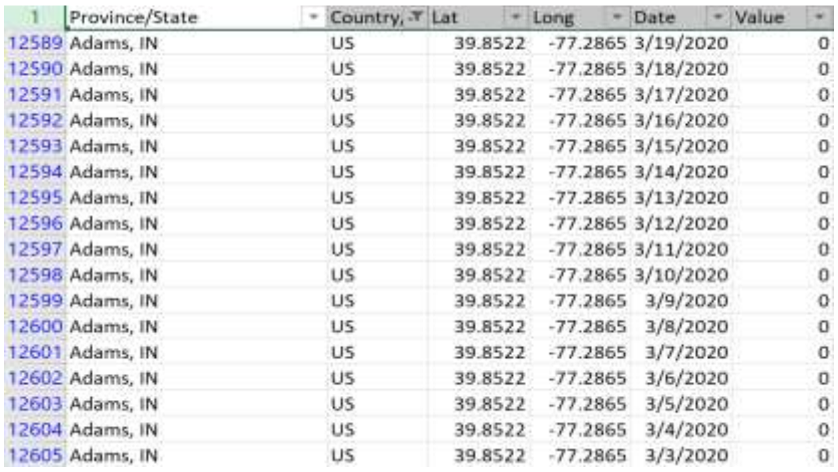
Data Cleaning / Scrubbing:

1. Initially, my dataset had many non-relevant information which was not in the scope of the project, I dropped them.



1	Province/State	Country/Region	Date	Value	New State	State + Date
2		Afghanistan	43909	22		43909
3		Afghanistan	43908	22		43908
4		Afghanistan	43907	22		43907
5		Afghanistan	43906	21		43906
6		Afghanistan	43905	16		43905
7		Afghanistan	43904	11		43904
8		Afghanistan	43903	7		43903
9		Afghanistan	43902	7		43902
10		Afghanistan	43901	7		43901
11		Afghanistan	43900	5		43900
12		Afghanistan	43899	4		43899

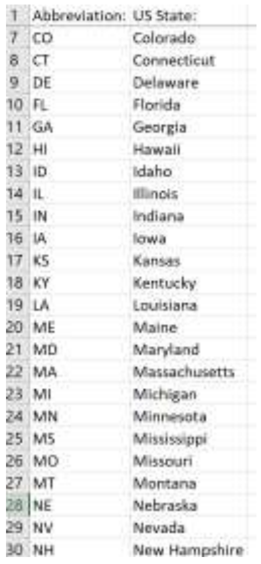
2. Filtered the Country to be USA, dropped the Lat and Long Columns



1	Province/State	Country/	Lat	Long	Date	Value
12589	Adams, IN	US	39.8522	-77.2865	3/19/2020	0
12590	Adams, IN	US	39.8522	-77.2865	3/18/2020	0
12591	Adams, IN	US	39.8522	-77.2865	3/17/2020	0
12592	Adams, IN	US	39.8522	-77.2865	3/16/2020	0
12593	Adams, IN	US	39.8522	-77.2865	3/15/2020	0
12594	Adams, IN	US	39.8522	-77.2865	3/14/2020	0
12595	Adams, IN	US	39.8522	-77.2865	3/13/2020	0
12596	Adams, IN	US	39.8522	-77.2865	3/12/2020	0
12597	Adams, IN	US	39.8522	-77.2865	3/11/2020	0
12598	Adams, IN	US	39.8522	-77.2865	3/10/2020	0
12599	Adams, IN	US	39.8522	-77.2865	3/9/2020	0
12600	Adams, IN	US	39.8522	-77.2865	3/8/2020	0
12601	Adams, IN	US	39.8522	-77.2865	3/7/2020	0
12602	Adams, IN	US	39.8522	-77.2865	3/6/2020	0
12603	Adams, IN	US	39.8522	-77.2865	3/5/2020	0
12604	Adams, IN	US	39.8522	-77.2865	3/4/2020	0
12605	Adams, IN	US	39.8522	-77.2865	3/3/2020	0

3. My dataset had state name along with the abbreviation, I created a new column for the states called New State where I cleaned it by adding a formula using excel and just obtained the state name and not the abbreviation.

4. I took the states and abbreviations from 50states.com where I could map it to a new sheet in my excel.



1	Abbreviation:	US State:
7	CO	Colorado
8	CT	Connecticut
9	DE	Delaware
10	FL	Florida
11	GA	Georgia
12	HI	Hawaii
13	ID	Idaho
14	IL	Illinois
15	IN	Indiana
16	IA	Iowa
17	KS	Kansas
18	KY	Kentucky
19	LA	Louisiana
20	ME	Maine
21	MD	Maryland
22	MA	Massachusetts
23	MI	Michigan
24	MN	Minnesota
25	MS	Mississippi
26	MO	Missouri
27	MT	Montana
28	NE	Nebraska
29	NV	Nevada
30	NH	New Hampshire

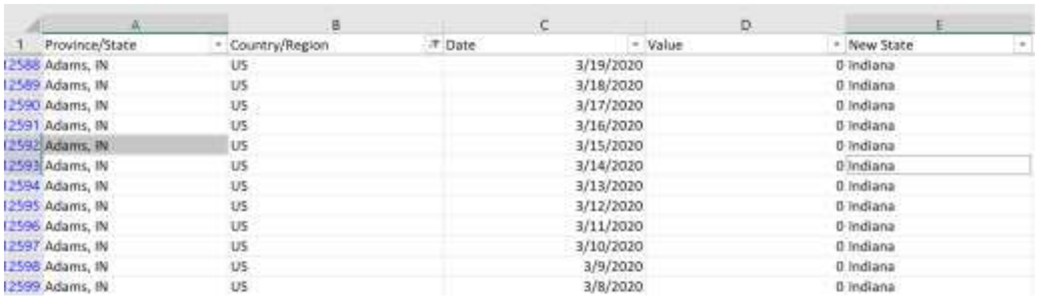
5. Below is the formula which I used to fetch the State Name

6. =IF(A2="","",IF(B2="US",IF(COUNTIF(A2,"*,*"),VLOOKUP(MID(A2,FIND(",",A2,1)+2, 2),States!A:B,2,FALSE),A2),A2))

7. The Vlookup is used when you need to find things in a table or a range by row. For example, look up a price of an automotive part by the part number, or find an employee name based on their employee ID.

8. This formula provides the new state name.

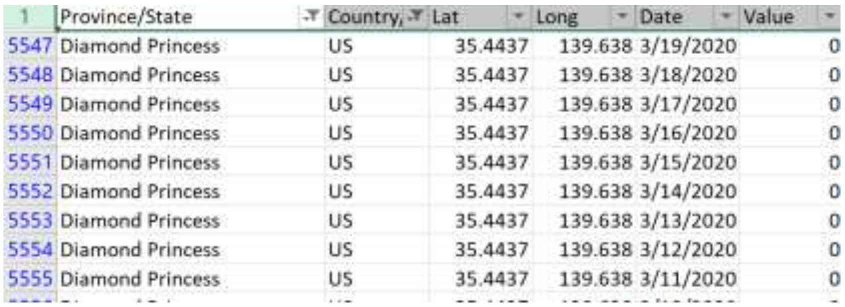
9. Below is the screenshot of the same.



	A	B	C	D	E
1	Province/State	Country/Region	Date	Value	New State
12588	Adams, IN	US	3/19/2020	0	Indiana
12589	Adams, IN	US	3/18/2020	0	Indiana
12590	Adams, IN	US	3/17/2020	0	Indiana
12591	Adams, IN	US	3/16/2020	0	Indiana
12592	Adams, IN	US	3/15/2020	0	Indiana
12593	Adams, IN	US	3/14/2020	0	Indiana
12594	Adams, IN	US	3/13/2020	0	Indiana
12595	Adams, IN	US	3/12/2020	0	Indiana
12596	Adams, IN	US	3/11/2020	0	Indiana
12597	Adams, IN	US	3/10/2020	0	Indiana
12598	Adams, IN	US	3/9/2020	0	Indiana
12599	Adams, IN	US	3/8/2020	0	Indiana

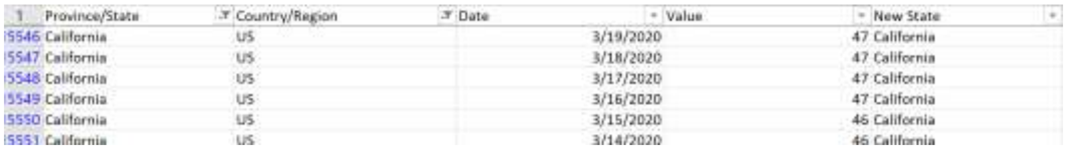
10. There were records of Diamond Princess cruise ship which were shown in records, I replaced them to the state where they belonged to, same way There were records of District of Columbia , I replaced them to Washington.

11. Before Cleaning



1	Province/State	Country/Region	Lat	Long	Date	Value
5547	Diamond Princess	US	35.4437	139.638	3/19/2020	0
5548	Diamond Princess	US	35.4437	139.638	3/18/2020	0
5549	Diamond Princess	US	35.4437	139.638	3/17/2020	0
5550	Diamond Princess	US	35.4437	139.638	3/16/2020	0
5551	Diamond Princess	US	35.4437	139.638	3/15/2020	0
5552	Diamond Princess	US	35.4437	139.638	3/14/2020	0
5553	Diamond Princess	US	35.4437	139.638	3/13/2020	0
5554	Diamond Princess	US	35.4437	139.638	3/12/2020	0
5555	Diamond Princess	US	35.4437	139.638	3/11/2020	0

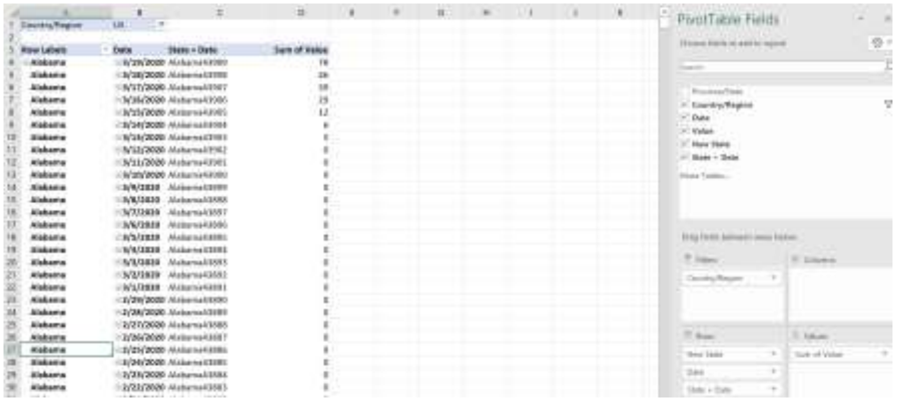
12. After Cleaning



1	Province/State	Country/Region	Date	Value	New State
5546	California	US	3/19/2020	47	California
5547	California	US	3/18/2020	47	California
5548	California	US	3/17/2020	47	California
5549	California	US	3/16/2020	47	California
5550	California	US	3/15/2020	46	California
5551	California	US	3/14/2020	46	California

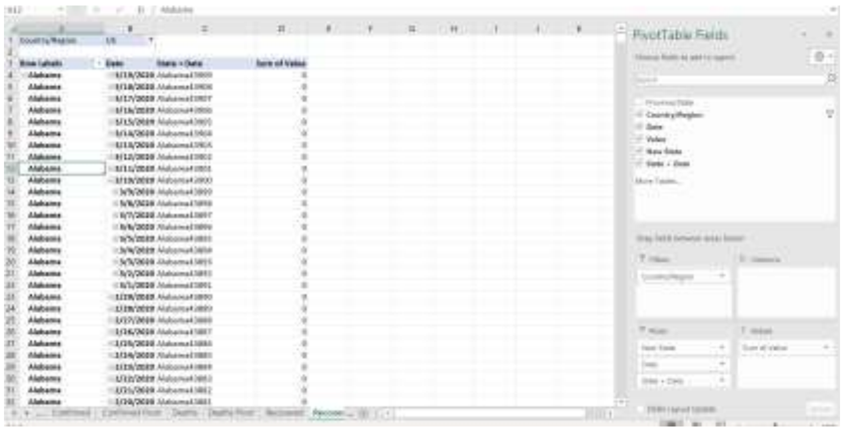
13. Created Pivot tables for Confirmed, Deaths and Recovered.

Confirmed_Pivot:



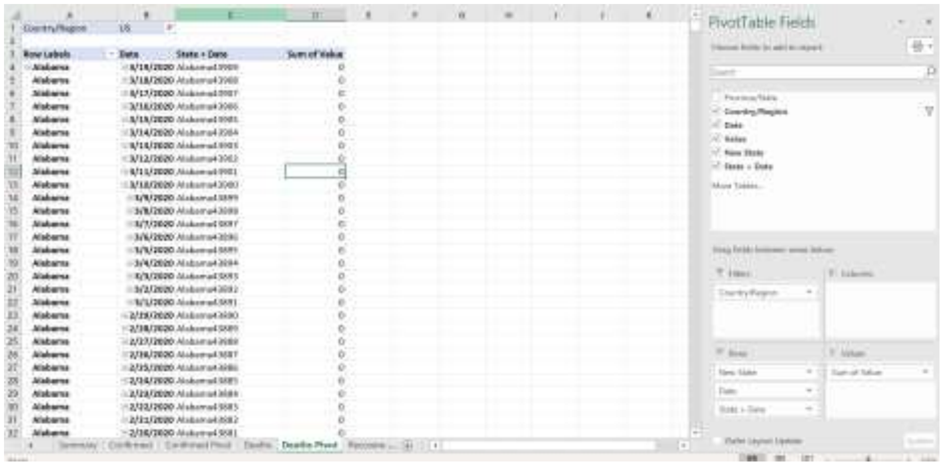
Country/Region	US		
Row Labels	Date	State + Date	Sum of Value
Alabama	3/19/2020	[illegible]	[illegible]
Alabama	3/18/2020	[illegible]	[illegible]
Alabama	3/17/2020	[illegible]	[illegible]
Alabama	3/16/2020	[illegible]	[illegible]
Alabama	3/15/2020	[illegible]	12
Alabama	3/14/2020	[illegible]	6
Alabama	3/13/2020	[illegible]	0
Alabama	3/12/2020	[illegible]	0
Alabama	3/11/2020	[illegible]	0
Alabama	3/10/2020	[illegible]	0
Alabama	3/9/2020	[illegible]	0
Alabama	3/8/2020	[illegible]	0
Alabama	3/7/2020	[illegible]	0
Alabama	3/6/2020	[illegible]	0
Alabama	3/5/2020	[illegible]	0
Alabama	3/4/2020	[illegible]	0
Alabama	3/3/2020	[illegible]	0
Alabama	3/2/2020	[illegible]	0
Alabama	3/1/2020	[illegible]	0
Alabama	2/29/2020	[illegible]	0
Alabama	2/28/2020	[illegible]	0
Alabama	2/27/2020	[illegible]	0
Alabama	2/26/2020	[illegible]	0
Alabama	2/25/2020	[illegible]	0
Alabama	2/24/2020	[illegible]	0
Alabama	2/23/2020	[illegible]	0
Alabama	2/22/2020	[illegible]	0

Recovered_Pivot



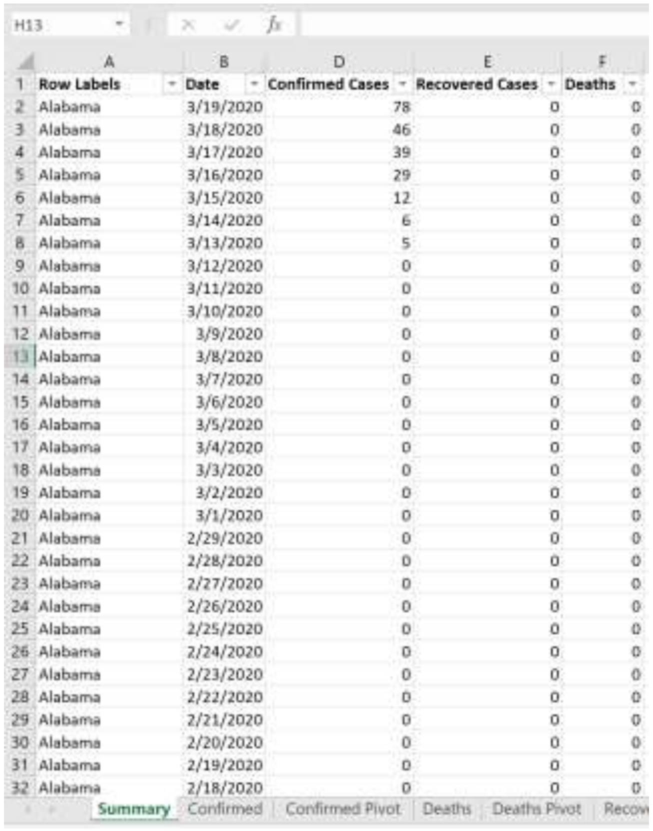
Country/Region	US		
Row Labels	Date	State + Date	Sum of Value
Alabama	3/19/2020	[illegible]	0
Alabama	3/18/2020	[illegible]	0
Alabama	3/17/2020	[illegible]	0
Alabama	3/16/2020	[illegible]	0
Alabama	3/15/2020	[illegible]	0
Alabama	3/14/2020	[illegible]	0
Alabama	3/13/2020	[illegible]	0
Alabama	3/12/2020	[illegible]	0
Alabama	3/11/2020	[illegible]	0
Alabama	3/10/2020	[illegible]	0
Alabama	3/9/2020	[illegible]	0
Alabama	3/8/2020	[illegible]	0
Alabama	3/7/2020	[illegible]	0
Alabama	3/6/2020	[illegible]	0
Alabama	3/5/2020	[illegible]	0
Alabama	3/4/2020	[illegible]	0
Alabama	3/3/2020	[illegible]	0
Alabama	3/2/2020	[illegible]	0
Alabama	3/1/2020	[illegible]	0
Alabama	2/29/2020	[illegible]	0
Alabama	2/28/2020	[illegible]	0
Alabama	2/27/2020	[illegible]	0
Alabama	2/26/2020	[illegible]	0
Alabama	2/25/2020	[illegible]	0
Alabama	2/24/2020	[illegible]	0
Alabama	2/23/2020	[illegible]	0
Alabama	2/22/2020	[illegible]	0
Alabama	2/21/2020	[illegible]	0
Alabama	2/20/2020	[illegible]	0

Deaths_Pivot



Country/Region	US		
Row Labels	Date	State + Date	Sum of Value
Alabama	3/19/2020	[illegible]	0
Alabama	3/18/2020	[illegible]	0
Alabama	3/17/2020	[illegible]	0
Alabama	3/16/2020	[illegible]	0
Alabama	3/15/2020	[illegible]	0
Alabama	3/14/2020	[illegible]	0
Alabama	3/13/2020	[illegible]	0
Alabama	3/12/2020	[illegible]	0
Alabama	3/11/2020	[illegible]	0
Alabama	3/10/2020	[illegible]	0
Alabama	3/9/2020	[illegible]	0
Alabama	3/8/2020	[illegible]	0
Alabama	3/7/2020	[illegible]	0
Alabama	3/6/2020	[illegible]	0
Alabama	3/5/2020	[illegible]	0
Alabama	3/4/2020	[illegible]	0
Alabama	3/3/2020	[illegible]	0
Alabama	3/2/2020	[illegible]	0
Alabama	3/1/2020	[illegible]	0
Alabama	2/29/2020	[illegible]	0
Alabama	2/28/2020	[illegible]	0
Alabama	2/27/2020	[illegible]	0
Alabama	2/26/2020	[illegible]	0
Alabama	2/25/2020	[illegible]	0
Alabama	2/24/2020	[illegible]	0
Alabama	2/23/2020	[illegible]	0
Alabama	2/22/2020	[illegible]	0
Alabama	2/21/2020	[illegible]	0
Alabama	2/18/2020	[illegible]	0

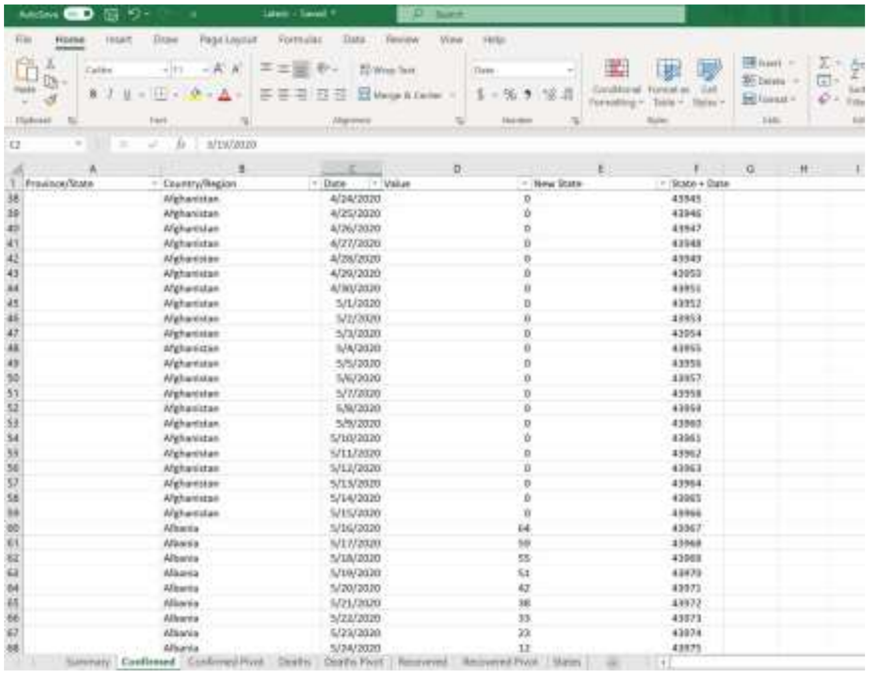
I also created a Summary of all the counts cumulative(Confirmed + Deaths + Recovered) into one excel sheet which will go to the tableau plugin directly.



Row Labels	Date	Confirmed Cases	Recovered Cases	Deaths
Alabama	3/19/2020	78	0	0
Alabama	3/18/2020	46	0	0
Alabama	3/17/2020	39	0	0
Alabama	3/16/2020	29	0	0
Alabama	3/15/2020	12	0	0
Alabama	3/14/2020	6	0	0
Alabama	3/13/2020	5	0	0
Alabama	3/12/2020	0	0	0
Alabama	3/11/2020	0	0	0
Alabama	3/10/2020	0	0	0
Alabama	3/9/2020	0	0	0
Alabama	3/8/2020	0	0	0
Alabama	3/7/2020	0	0	0
Alabama	3/6/2020	0	0	0
Alabama	3/5/2020	0	0	0
Alabama	3/4/2020	0	0	0
Alabama	3/3/2020	0	0	0
Alabama	3/2/2020	0	0	0
Alabama	3/1/2020	0	0	0
Alabama	2/29/2020	0	0	0
Alabama	2/28/2020	0	0	0
Alabama	2/27/2020	0	0	0
Alabama	2/26/2020	0	0	0
Alabama	2/25/2020	0	0	0
Alabama	2/24/2020	0	0	0
Alabama	2/23/2020	0	0	0
Alabama	2/22/2020	0	0	0
Alabama	2/21/2020	0	0	0
Alabama	2/20/2020	0	0	0
Alabama	2/19/2020	0	0	0
Alabama	2/18/2020	0	0	0

I chose Excel over scala because:

1. The Date Format was not clear in the csv. I got junk date column values and invalid dates.



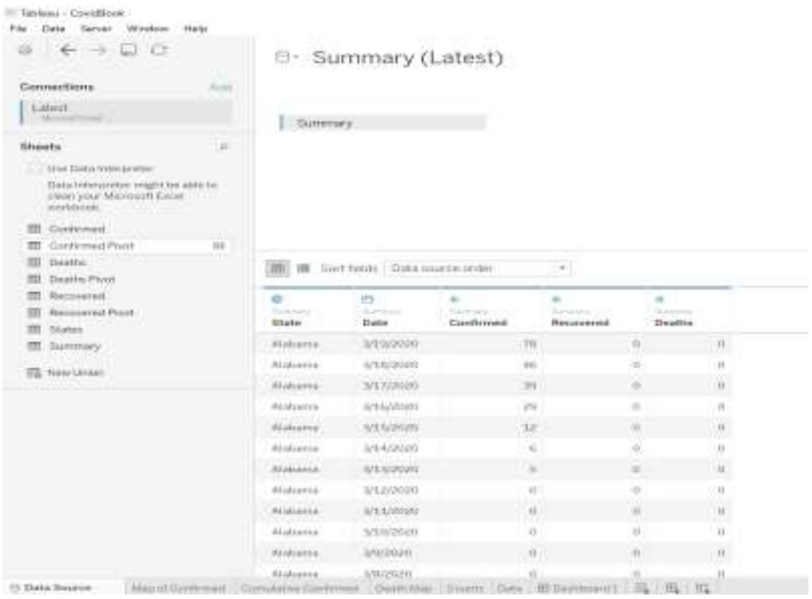
	Province/State	Country/Region	Date	Value	New State	State + Date
38		Afghanistan	4/24/2020	0		43945
39		Afghanistan	4/25/2020	0		43946
40		Afghanistan	4/26/2020	0		43947
41		Afghanistan	4/27/2020	0		43948
42		Afghanistan	4/28/2020	0		43949
43		Afghanistan	4/29/2020	0		43950
44		Afghanistan	4/30/2020	0		43951
45		Afghanistan	5/1/2020	0		43952
46		Afghanistan	5/2/2020	0		43953
47		Afghanistan	5/3/2020	0		43954
48		Afghanistan	5/4/2020	0		43955
49		Afghanistan	5/5/2020	0		43956
50		Afghanistan	5/6/2020	0		43957
51		Afghanistan	5/7/2020	0		43958
52		Afghanistan	5/8/2020	0		43959
53		Afghanistan	5/9/2020	0		43960
54		Afghanistan	5/10/2020	0		43961
55		Afghanistan	5/11/2020	0		43962
56		Afghanistan	5/12/2020	0		43963
57		Afghanistan	5/13/2020	0		43964
58		Afghanistan	5/14/2020	0		43965
59		Afghanistan	5/15/2020	0		43966
60		Albania	5/16/2020	64		43967
61		Albania	5/17/2020	59		43968
62		Albania	5/18/2020	55		43969
63		Albania	5/19/2020	51		43970
64		Albania	5/20/2020	42		43971
65		Albania	5/21/2020	38		43972
66		Albania	5/22/2020	33		43973
67		Albania	5/23/2020	23		43974
68		Albania	5/24/2020	12		43975

We see a lot of invalid dates.

2. My end goal was to do Visualization using tableau; hence Excel would be very compatible its also something that I have previous experience working with it.

Visualization on Tableau

Initial Setup

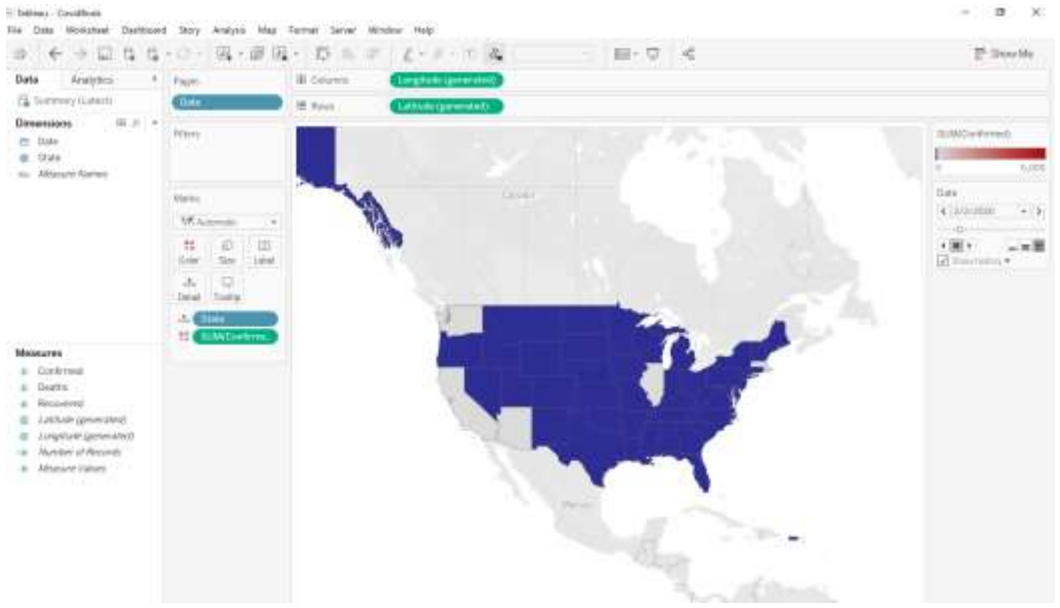


Summary (Latest)

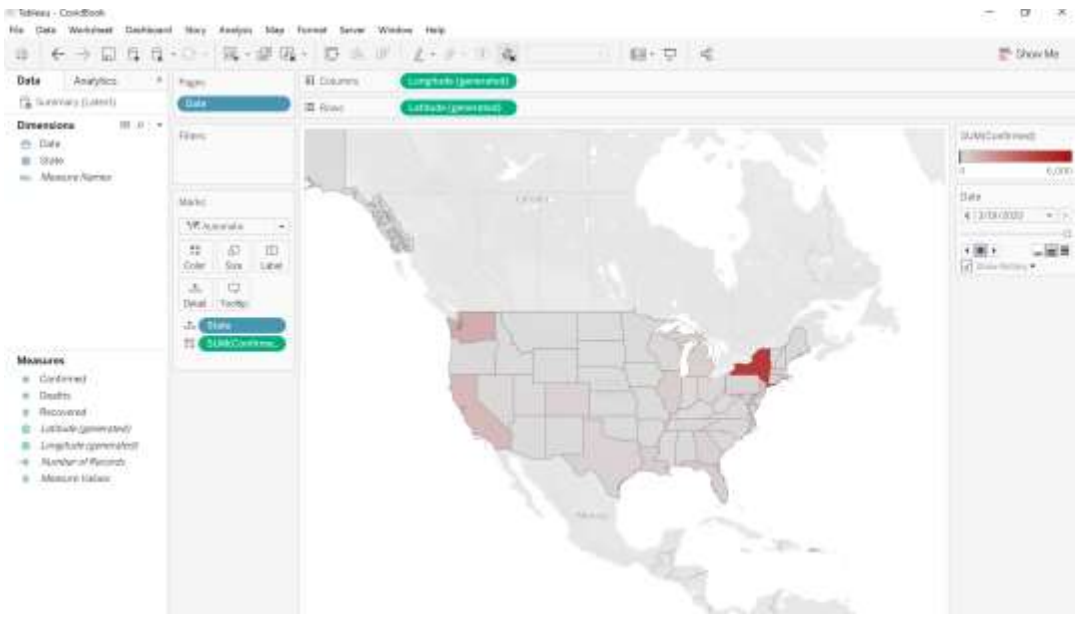
State	Date	Confirmed	Recovered	Deaths
Alabama	3/19/2020	78	0	0
Alabama	3/18/2020	46	0	0
Alabama	3/17/2020	39	0	0
Alabama	3/16/2020	29	0	0
Alabama	3/15/2020	12	0	0
Alabama	3/14/2020	6	0	0
Alabama	3/13/2020	5	0	0
Alabama	3/12/2020	0	0	0
Alabama	3/11/2020	0	0	0
Alabama	3/10/2020	0	0	0
Alabama	3/9/2020	0	0	0
Alabama	3/8/2020	0	0	0

Visualization 1 : Time series of outbreak of COVID- 19 pandemic in the US, which happened on 1/22/2020 till the National Emergency declaration on the 18th of Feb.

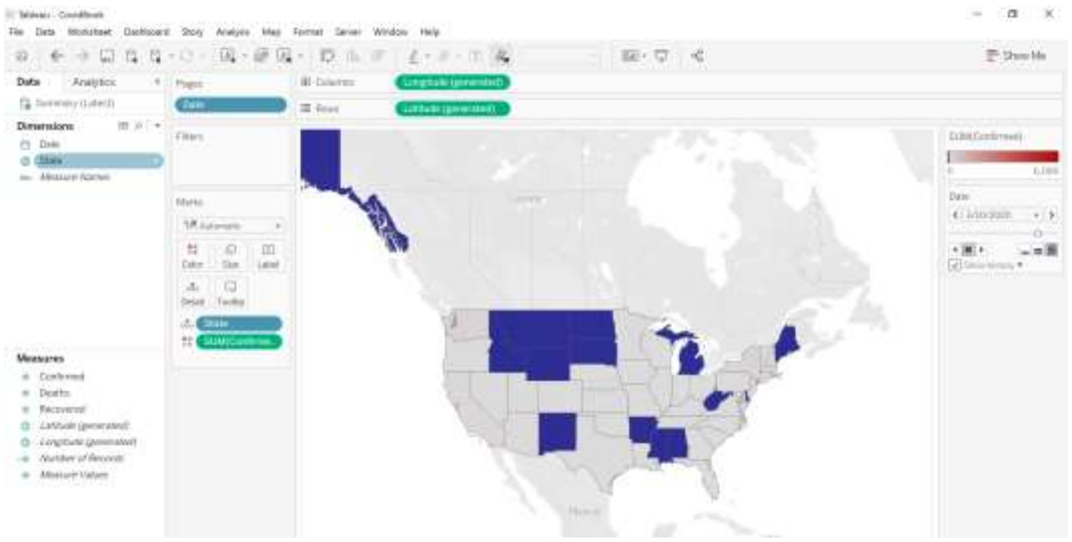
Visualization as on 2/2/2020



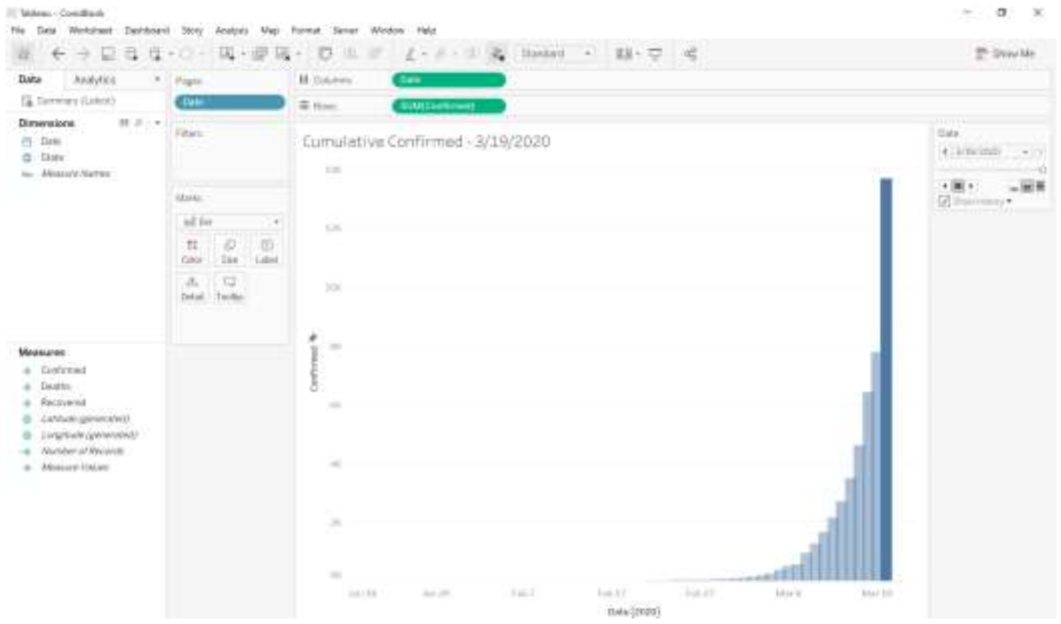
Visualization as on 3/19/2020, where we can see that New York has the highest number of cases.



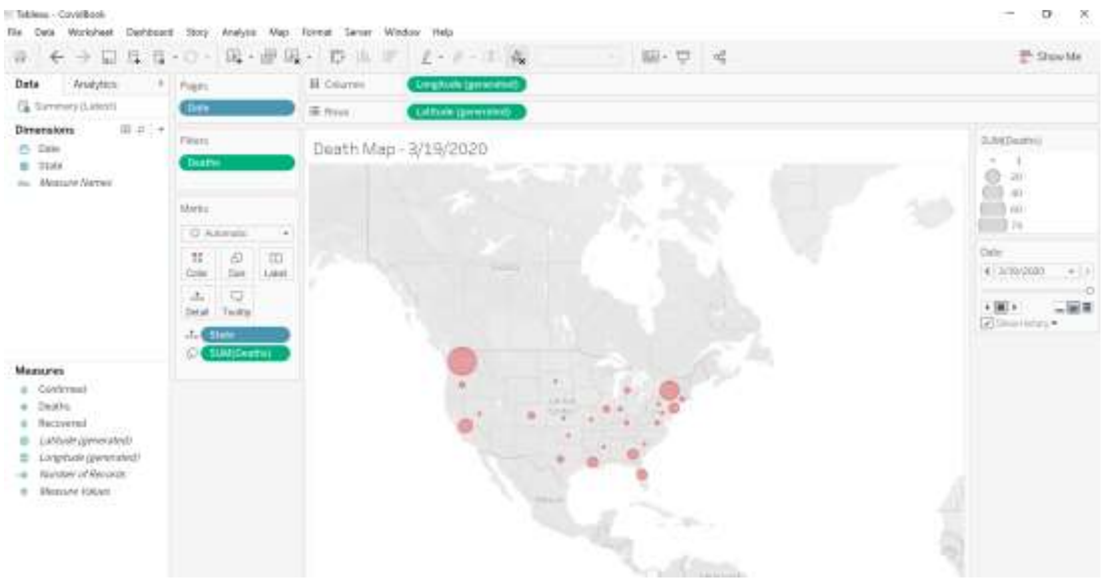
Visualization as on 3/10/2020: We see a few states which are still not badly affected by the pandemic.



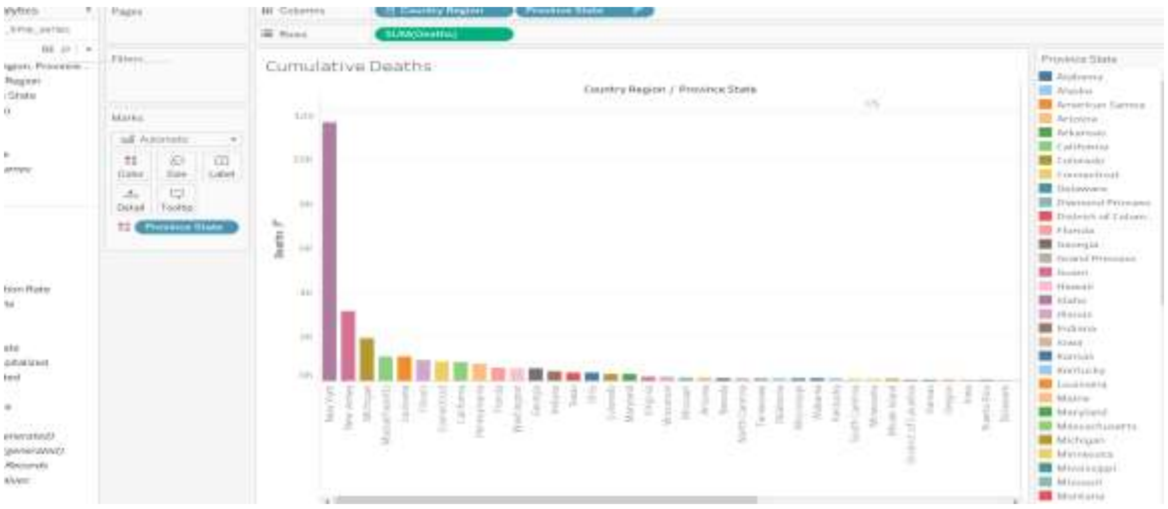
Visualization 2: To take the total number of cases. Visualization as on 3/19/2020



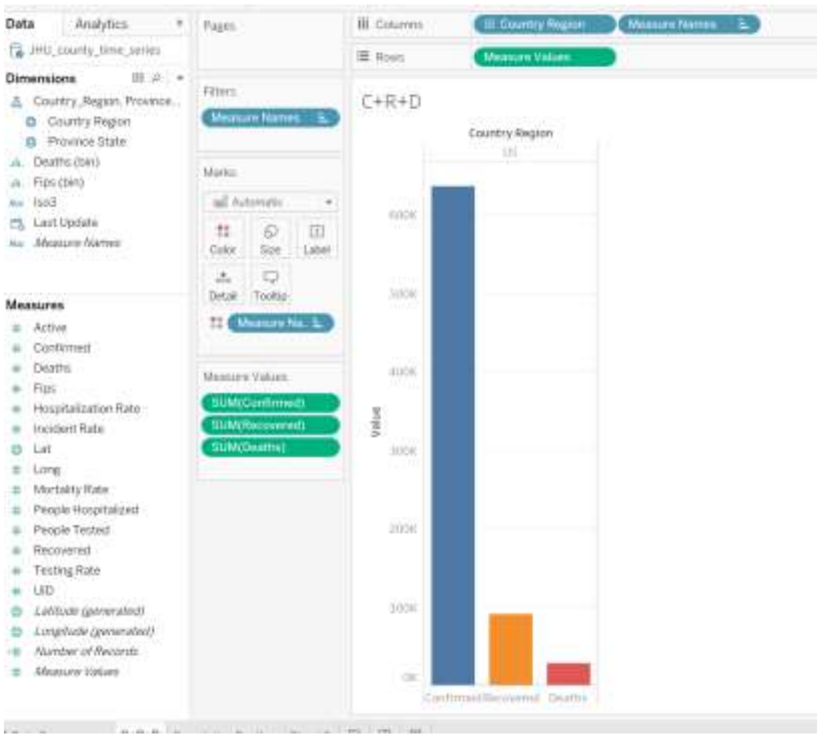
Visualization 3: Death Map : Showing the number of deaths in different states of the USA.



Visualization 4: Cumulative Deaths counts in different states.



Visualization 5: Summary of all cases.



Columns: Country Region, Measure Names

Rows: Measure Values

Filters: Measure Names

Measure Values: SUM(Confirmed), SUM(Recovered), SUM(Deaths)

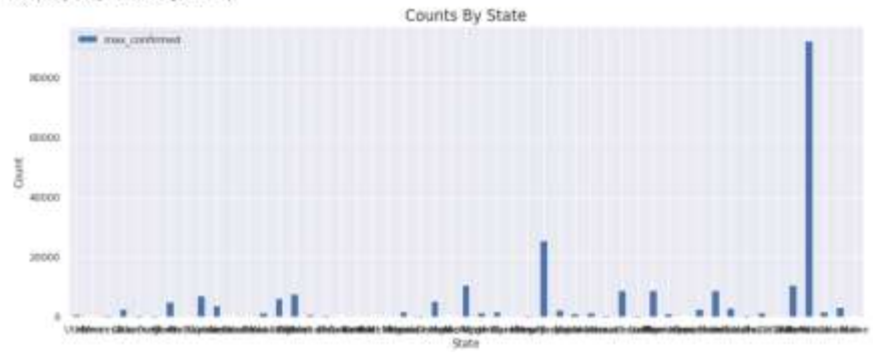
Additional Submission

1. I did an additional visualization on a dataset which was available in our project GitHub repository. (WorldCapitals/COVID-19). It was available in a form of a csv I imported the CSV.
2. Performed Data Cleaning.
3. Created a dataframe using pyspark.
4. Performed SQL queries on the temp tables using pyspark
5. Performed data analysis using pyspark.
6. Visualization.

Below are a few visualizations which I obtained

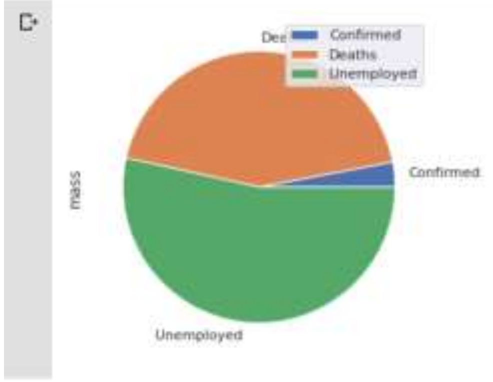
Visualization 1: Counts of Confirmed cases in each state.

```
Text(0.5, 1.0, 'Counts By State')
```



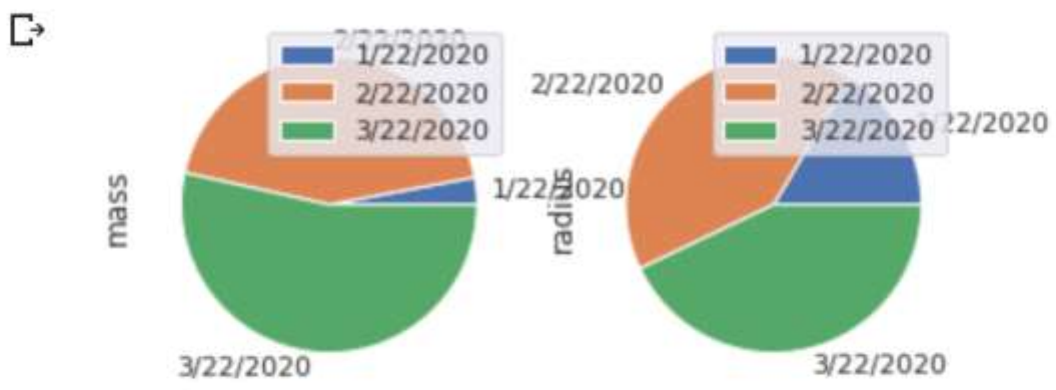
Visualization 2: Pie Chart of Confirmed count, deaths and unemployment.

```
[ ] df_sp = pd.DataFrame({'mass': [0.330, 4.87 , 5.97],
                         'radius': [2439.7, 6051.8, 6378.1]},
                        index=['Confirmed', 'Deaths', 'Unemployed'])
    plot = df_sp.plot.pie(y='mass', figsize=(5, 5))
```

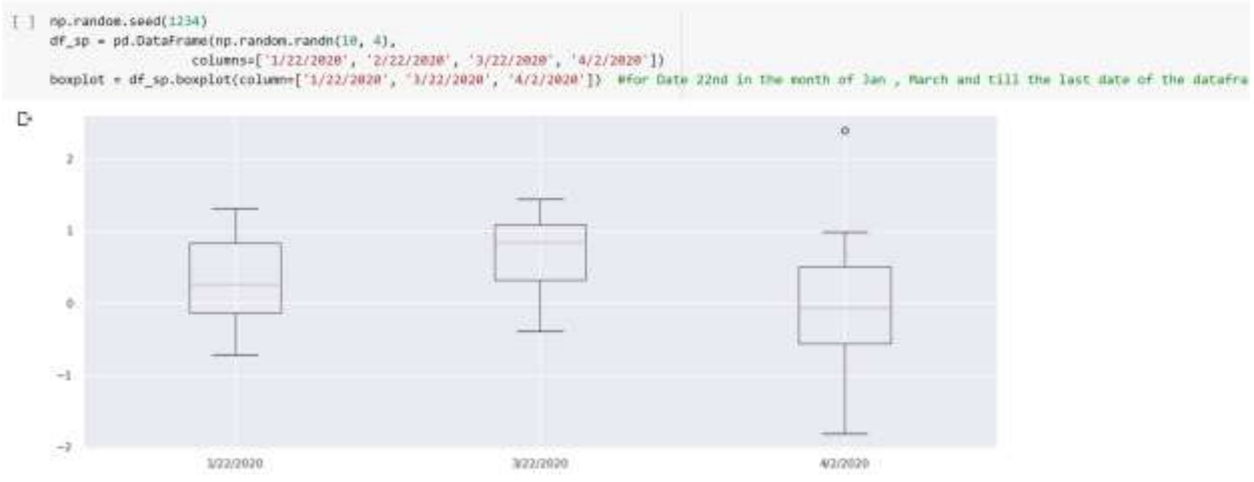


Visualization 3: Pandemic cases on 1/22/20, 2/22/20, 3/22/20.

```
[ ] plot = df_sp.plot.pie(subplots=True, figsize=(6, 3))
```

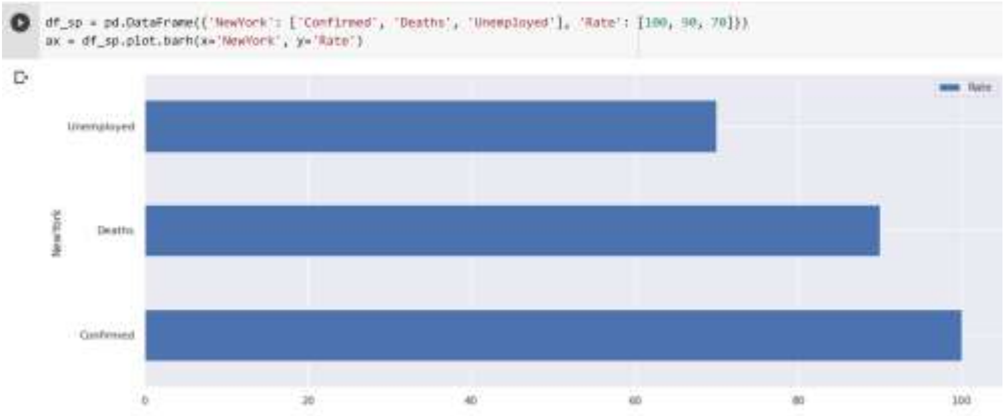


Visualization 4: Pandemic cases on 1/22/20, 2/22/20, 3/22/20.



```
[ ] np.random.seed(1234)
    df_sp = pd.DataFrame(np.random.randn(10, 4),
                      columns=['1/22/2020', '2/22/2020', '3/22/2020', '4/2/2020'])
    boxplot = df_sp.boxplot(column=['1/22/2020', '3/22/2020', '4/2/2020']) #for Date 22nd in the month of Jan , March and till the last date of the datafra
```

Visualization 5: Scenario in New York was what plotted by making analysis.



```
df_sp = pd.DataFrame({'NewYork': ['Confirmed', 'Deaths', 'Unemployed'], 'Rate': [100, 90, 70]})
ax = df_sp.plot.barh(x='NewYork', y='Rate')
```

Tableau Dashboard

I collaborated with Kyle to design the dashboard, where we combined both our visualizations and formed a tableau dashboard. Kyle has already added the visualization screenshots. We could enhance the visualizations and improve the look and feel of the dashboard.

Project Experience

I found it quite challenging to collaborate with the group with the ongoing lockdowns and no face to face interactions . Also, the group used MMS for communicating and updating regarding the project , where my phone had an issue receiving those messages and knowing about the project updates. Also, it was very hard to reach every member and interact about the project. Only Kyle was easily accessible. My major collaboration was mainly with Kyle.

Project Conclusion

In conclusion, the project accomplished most of the Project Schedule and Project Objectives. All the components from the Project Objective were either individually complete with the only step being complete integration. The Project Schedule implemented a shortened Agile framework focusing on a condensed version based on industry standards. Following this framework, we decided on 2 (1) week machine learning model iterations, followed by a week of pipeline, and a week of dashboard integration. Ultimately, the project ran out of time needing one or two more weeks for project completion. The pipeline created operates using D3 plotting top 10 confirmed countries and running on a Kafka server locally. Several models were built but the AutoML GBM Multinomial model was the top performer with a 0% training error, 99.81% R-Square, 67.69% MSE, and 82.16% RSME. Stunning visualizations using heatmaps and dashboard was developed using Tableau. It is worth noting that Michigan has the highest Mortality Rate (8.2) in all the country.