Ismaeel AlAlawi
BU ID: U29987528
Lab Section: C1
EK125
Spring 2020

Final Project: Machine Learning & the COVID-19 Outbreak Apex

Over the past few decades, data science has revolutionized the way we look at data in a multitude of fields and has helped us reach new heights in technological innovation. There are various techniques employed by data analysts to make sense of huge piles of data. Chief among these techniques are feature engineering and statistical analysis. Feature engineering allows data scientists to 'scrub' the data into useful, relevant, less unbiased data where redundancies are eliminated and missing data are either ignored, removed, or replaced with mean values. Upon scrubbing the data, statistical tools can be used to extract meaningful trends from the data. These two tools lay the foundation for another major tool called machine learning, a subset of artificial intelligence, which involves machines learning from datasets without explicit programming and making predictions with an eventual goal of maximizing task performance and accuracy. There are two main machine learning tools: classification and regression. Within these two sets of tools are various ML algorithms which will be discussed in the ML Toolbox section found below.

Even though machine learning trained models can make fairly accurate predictions about data, there lie **intrinsic biases in the trained models** due to the selection of the training datasets used in creating the models. According to the Founder of Lips[1], Annie Brown, who writes in the *Forbes* article *Biased Algorithms Learn From Biased Data: 3 Kinds Biases Found In AI Datasets, "*Imagine a scenario in which self-driving cars fail to recognize people of color as people – and are thus more likely to hit them – because the computers were trained on data sets of photos in which such people were absent or underrepresented" (**article link found in references and footnote below**). Brown underlines that recent research has revealed that facial recognition software from big tech companies like Microsoft and Amazon are able to identify white males but not darker-skinned women. Brown then proceeds to explain that this discrepancy happens due to algorithmic bias and can be classified into three different types: *interaction bias*, *latent bias*, and *selection bias*. For interaction bias, Brown crystallizes that happens when algorithms are trained using datasets that are intrinsically biased by how users interact with it; a typical example is the aforementioned facial recognition software bias. For latent bias, Brown underlines that an algorithm may inaccurately classify/identify objects based on historical data or inherent societal stereotypes. Lastly, for selection bias, datasets are overrepresented by certain groups while underrepresented by others. In conclusion, Brown tackles modern solutions and efforts to combat algorithmic bias like IBM's AI Fairness 360, an open source tool that can detect bias in datasets, and Facebook AI's new image classification techniques that marks dataset images that the model used to train; Brown calls these solutions, or verification methods, as "radioactive data."

One of the most pressing societal problems everyone's facing today is the **outbreak of the COVID-19 virus**. With the majority of people having self-quarantined themselves to not expose

[1] Lips is an AI and blockchain-enabled social commerce platform for women and LGBTQ creators. Article link: https://www.forbes.com/sites/cognitiveworld/2020/02/07/biased-algorithms/#63bfdc5576fc

themselves or others to risk, several questions arise. Namely, when will we return to our normal pre-outbreak lives? Who is more likely to be at risk of contracting the virus? Which countries seem to be heading more closely towards an apex and why? How comparable are COVID-19 'apex' trends to the Swine Flu, and the deadly 1918 Influenza? How has the shift to self-quarantine affected various economies and industries? How will it continue to affect them? How much shortage of supplies is there in dealing with COVID-19? These are important questions that will be answered by using the MATLAB Machine Learning Toolbox. This is evidently meaningful to society as it will help alleviate some of our current concerns regarding the continuity of the virus, its ability to spread to loved ones, and its impact on us. The virus has affected both our mental and physical well-being, so it is certainly in our best interest to intelligently guess when it will end using ML algorithms!

As it is difficult to come by reliable data sets that would permit us to answer all the questions asked (especially since the outbreak is a recent occurrence), then we should sharpen our focus towards one big question that matters the most. For this project, the big question is: when is the outbreak projected to end? This is the most important question because a lot of people have been concerned regarding the end of the outbreak and its current (as well as lasting) effects on our social dynamics, economy, and so on; in fact, it has even changed our inherent culture of giving hugs and kisses when greeting our loved ones! In order to find this apex date, we need to look at the progression of the confirmed number of cases relative to time (or the dates, in this case). Thus, I use a large 102 x 265 data set (27,030 data points) which lists the time series of the global confirmed cases (country-by country). This is supplied by the John Hopkins School of Public Health database via: https://data.humdata.org/dataset/novel-coronavirus-2019-ncov-cases. The file is called ***time_series_covid19_confirmed_global.csv***; fortunately, it's in .csv format, so it can easily be imported into MATLAB.

Feature Engineering & Data Scrubbing

Feature Redundancy and Removal Technique

After importing the COVID-19 global confirmed cases table into MATLAB, I noticed that some of the feature engineering techniques we learned in Chapter 14A should be employed before using the Machine Learning Toolbox. After importing the table, we show the first eight rows and six columns of the data set.

```
>> dataset = readtable('time_series_covid19_confirmed_global.csv');
>> head(dataset)

ans =

  8×102 table
```

Var1	Var2	Var3	Var4	Var5	Var6
'Province/State'	'Country/Region'	'Lat'	'Long'	'1/22/20'	'1/23/20'
''	'Afghanistan'	'33.0'	'65.0'	'0'	'0'
''	'Albania'	'41.1533'	'20.1683'	'0'	'0'
''	'Algeria'	'28.0339'	'1.6596'	'0'	'0'
''	'Andorra'	'42.5063'	'1.5218'	'0'	'0'
''	'Angola'	'-11.2027'	'17.8739'	'0'	'0'
''	'Antigua and Barbuda'	'17.0608'	'-61.7964'	'0'	'0'
''	'Argentina'	'-38.4161'	'-63.6167'	'0'	'0'

Evidently, the column names are reported as elements in the table, and the column names themselves are denoted with some arbitrary name 'Var.' Thus, in order to fix this, we re-assign the first row of the table to the table column names, and then delete the first row. It is important to note that we have to use the table2cell() MATLAB built-in function in order to convert the table row into a cell array as required.

```
>> dataset.Properties.VariableNames = table2cell(dataset(1,:))
'Province/State' is not a valid table variable name. See the documentation for isvarname or
matlab.lang.makeValidName for more information.
```

However, we run into an error when we do this. As noted above, this is because the '/' is not allowed in a variable name, since according to MATLAB documentation, a valid variable name begins with a letter and can include letters, digits, and underscores. We can see that '/' is used for the dates of the confirmed cases and the first two columns, so we would can alternatively replace the '/' with an underscore! We can do this easily using two for loops:

```
[r,c] = size(dataset); % grab rows and columns of dataset

for i = 1:r % loop through table to index content
    for j = 1:c
        if ismember('/', dataset{i,j}{1}) % if '/' exists in element
            dataset{i,j}{1} = strrep(dataset{i,j}{1}, '/', '_'); % replace '/'
        end
    end
end
```

We show this change again by using head():

Var1	Var2	Var3	Var4	Var5	Var6
'Province_State'	'Country_Region'	'Lat'	'Long'	'1_22_20'	'1_23_20'
''	'Afghanistan'	'33.0'	'65.0'	'0'	'0'
''	'Albania'	'41.1533'	'20.1683'	'0'	'0'
''	'Algeria'	'28.0339'	'1.6596'	'0'	'0'
''	'Andorra'	'42.5063'	'1.5218'	'0'	'0'
''	'Angola'	'-11.2027'	'17.8739'	'0'	'0'
''	'Antigua and Barbuda'	'17.0608'	'-61.7964'	'0'	'0'
''	'Argentina'	'-38.4161'	'-63.6167'	'0'	'0'

Now, we notice that the date labels start with a digit, not a letter as required by MATLAB. So we can add a letter 'D' in order to indicate that's the date; we do this by horizontal concatenation in a loop:

```
for j = 5:c
    dataset{1,j}{1} = ['D' dataset{1,j}{1}];
end
```

Then, we rename the column names and delete the first row:

```
dataset.Properties.VariableNames = dataset{1,:}; % rename table columns
dataset(1,:) = []; % delete first row
```

After that, we get:

Province_State	Country_Region	Lat	Long	D1_22_20	D1_23_20	D1_24_20
''	'Afghanistan'	'33.0'	'65.0'	'0'	'0'	'0'
''	'Albania'	'41.1533'	'20.1683'	'0'	'0'	'0'
''	'Algeria'	'28.0339'	'1.6596'	'0'	'0'	'0'
''	'Andorra'	'42.5063'	'1.5218'	'0'	'0'	'0'
''	'Angola'	'-11.2027'	'17.8739'	'0'	'0'	'0'
''	'Antigua and Barbuda'	'17.0608'	'-61.7964'	'0'	'0'	'0'
''	'Argentina'	'-38.4161'	'-63.6167'	'0'	'0'	'0'
''	'Armenia'	'40.0691'	'45.0382'	'0'	'0'	'0'

With this more polished table, there is still room for improvement. Since we're only interested in countries and not regions within countries, we can rename the *Country_Region* column to Country. With this, we can remove the *Province_State* column. Additionally, since we are already given the country name, then it's *redundant* and not quite useful to have the Latitude and Longitude columns, so we shall delete those as well. We do this by executing the following code:

```
% Further Deletion and Renaming

dataset(:,1) = []; % delete Province_State column
dataset.Properties.VariableNames{'Country_Region'} = 'Country';
dataset(:,2:3) = [];
```

This yields:

Country	D1_22_20	D1_23_20	D1_24_20	D1_25_20	D1_26_20	D1_27_20
'Afghanistan'	'0'	'0'	'0'	'0'	'0'	'0'
'Albania'	'0'	'0'	'0'	'0'	'0'	'0'
'Algeria'	'0'	'0'	'0'	'0'	'0'	'0'
'Andorra'	'0'	'0'	'0'	'0'	'0'	'0'
'Angola'	'0'	'0'	'0'	'0'	'0'	'0'
'Antigua and Barbuda'	'0'	'0'	'0'	'0'	'0'	'0'
'Argentina'	'0'	'0'	'0'	'0'	'0'	'0'
'Armenia'	'0'	'0'	'0'	'0'	'0'	'0'

We should also arrange the country names alphabetically, since they're scrambled in the dataset. We can do this by using the sort() function, as follows:

```
% Alphabetic Arrangement of Country Names

dataset{:,1} = sort(dataset{:,1});
```

We can also remove redundant country names to make the table more concise:

```
% Country Redundancy Removal

dataset_machine_country = dataset;

% Assign sum of redundant features for given country to a single feature
dataset_machine_country{9,2:end} = num2cell(sum(cell2mat(dataset{9:16, 2:end})));
dataset_machine_country{40,2:end} = num2cell(sum(cell2mat(dataset{40:54, 2:end})));
dataset_machine_country{58,2:end} = num2cell(sum(cell2mat(dataset{58:90, 2:end})));
dataset_machine_country{92,2:end} = num2cell(sum(cell2mat(dataset{92:93, 2:end})));
dataset_machine_country{100,2:end} = num2cell(sum(cell2mat(dataset{100:102, 2:end})));
dataset_machine_country{117,2:end} = num2cell(sum(cell2mat(dataset{117:127, 2:end})));
dataset_machine_country{186,2:end} = num2cell(sum(cell2mat(dataset{186:190, 2:end})));
dataset_machine_country{245,2:end} = num2cell(sum(cell2mat(dataset{245:255, 2:end})));

% Delete redundant features
dataset_machine_country(10:16,:) = [];
dataset_machine_country(34:47,:) = [];
dataset_machine_country(38:69,:) = [];
dataset_machine_country(40,:) = [];
dataset_machine_country(47:48,:) = [];
dataset_machine_country(62:71,:) = [];
dataset_machine_country(121:124,:) = [];
dataset_machine_country(176:185,:) = [];
```

Normalization Technique and Dimensionality Reduction for Machine Learning Model

For the general machine learning model, in which we want to look at us the cases globally, we can normalize the data and reduce the dimensions of the dataset since the model can only train using numerical data (or predictors). Thus, we develop a new metric assignment, in which January 22nd, 2020 (which is column D1_22_20) is translated into the number 1, and 1 is added to each consecutive day afterwards, i.e. D1_23_20 would be 2, D1_24_20 is 3, and so on. This way, the machine learning model will be able to train using the numerical translation of the dates in order to output a numerical number which we would have to translate back to the equivalent date. Secondly, it can be established that the number of confirmed cases is reported as a character in each field, so we need to use str2double() to convert the cases into real numbers. After this translation, we can create a new table which takes the sum of all the cases globally for each given day. Note that we do this only for our global COVID-19 analysis, and not the country-by-country COVID-19 analysis. We apply the following the following code to accomplish the second task:

```
[r,c] = size(dataset); % grab new rows and new columns of dataset

for i = 1:r
    for j = 2:c
        dataset{i,j}{1} = str2double(dataset{i,j}{1}); % convert to double
    end
end
```

Country	D1_22_20	D1_23_20	D1_24_20	D1_25_20	D1_26_20	D1_27_20
'Afghanistan'	[0]	[0]	[0]	[0]	[0]	[0]
'Albania'	[0]	[0]	[0]	[0]	[0]	[0]
'Algeria'	[0]	[0]	[0]	[0]	[0]	[0]
'Andorra'	[0]	[0]	[0]	[0]	[0]	[0]
'Angola'	[0]	[0]	[0]	[0]	[0]	[0]
'Antigua and Barbuda'	[0]	[0]	[0]	[0]	[0]	[0]
'Argentina'	[0]	[0]	[0]	[0]	[0]	[0]
'Armenia'	[0]	[0]	[0]	[0]	[0]	[0]

Note that the entries are in square brackets to indicate that they are scalar real numbers. The first task is a bit more complicated code-wise, and is given as follows:

```
dataset_machine = table2cell(dataset);

for j = 2:c
    dataset_machine{1,j} = j - 1; % numerical translation of dates
end
```

```
>> disp(dataset_machine(1:8,:))
  Columns 1 through 12

    'Afghanistan'           [1]    [2]    [3]    [4]    [5]    [6]    [7]    [8]    [9]    [10]    [11]
    'Albania'               [0]    [0]    [0]    [0]    [0]    [0]    [0]    [0]    [0]    [ 0]    [ 0]
    'Algeria'               [0]    [0]    [0]    [0]    [0]    [0]    [0]    [0]    [0]    [ 0]    [ 0]
    'Andorra'               [0]    [0]    [0]    [0]    [0]    [0]    [0]    [0]    [0]    [ 0]    [ 0]
    'Angola'                [0]    [0]    [0]    [0]    [0]    [0]    [0]    [0]    [0]    [ 0]    [ 0]
    'Antigua and Barbuda'   [0]    [0]    [0]    [0]    [0]    [0]    [0]    [0]    [0]    [ 0]    [ 0]
    'Argentina'             [0]    [0]    [0]    [0]    [0]    [0]    [0]    [0]    [0]    [ 0]    [ 0]
    'Armenia'               [0]    [0]    [0]    [0]    [0]    [0]    [0]    [0]    [0]    [ 0]    [ 0]
```

As you can see, this new dataset (which is now a cell array) includes numerical data for the dates, according to our conversion metric. For global analysis, it would behoove us to take the sum of all the country confirmed cases (for each day), and report it in a matrix which vertically concatenates the numerical data array with the sums for each day top to down. We accomplish this by doing:

```
dataset_machine_global = cell2mat(dataset_machine(:, 2:end));
dataset_machine_global = [dataset_machine_global(1,:);...
    sum(dataset_machine_global(2:end,:))]; % concatenate vectors
```

We'll display the first and last columns of this matrix (which represent how COVID-19 global confirmed cases have progressed, by the way):

```
>> disp([dataset_machine_global(:,1) dataset_machine_global(:,end)])
           1          98
         555     3114570
```

Yikes! This shows that on January 22nd , the total number of global confirmed cases increased by 5612 times so far as of April 28th ! In other words, it increases from 555 cases to 3 million cases worldwide.

Fortunately, we do not need to remove data points or analyze outliers. This is because the John Hopkins dataset provided doesn't contain missing data nor outliers.

Statistical Analysis

Correlation Between Features

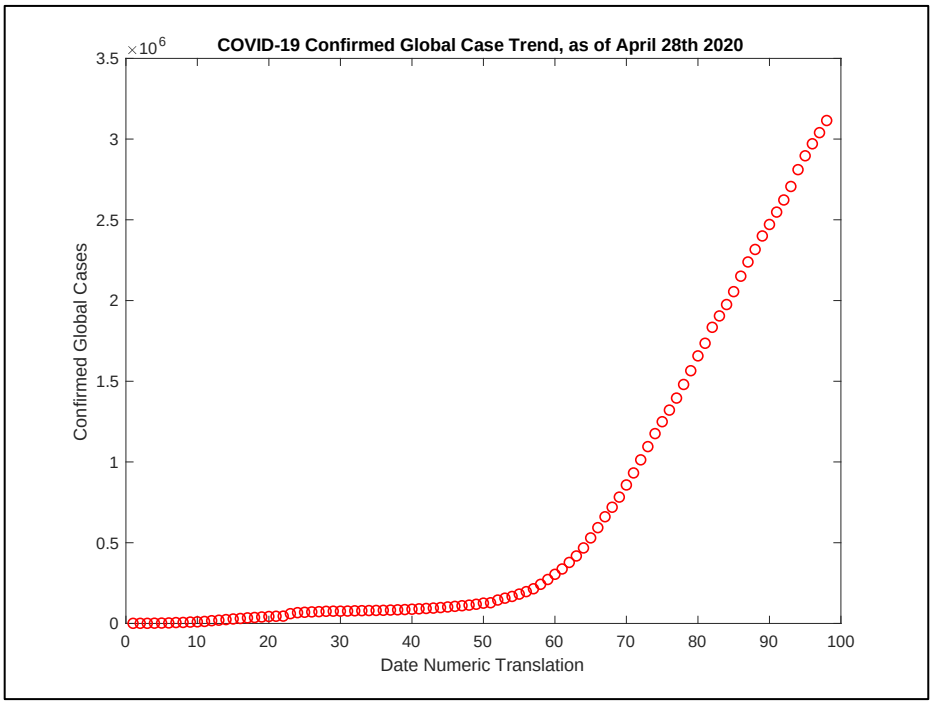
Before creating a machine learning model, it is important to understand the inherent statistical properties of our dataset. We'll only look at the global cases as it was reported recently on the dataset website, that unfortunately there are several discrepancies for the country-by-country data, yet the sums for each day are roughly accurate. In particular, we can analyze the correlation between the time that has passed and the number of COVID-19 cases. We do this as follows:

```
% Correlation Analysis
X_global = dataset_machine_global(1,:); % store numeric dates
Y_global = dataset_machine_global(2,:); % store global confirmed cases

coff = corrcoef(X_global, Y_global); % calculate correlation coefficient
```

```
coff =

    1.0000    0.8529
    0.8529    1.0000
```

As it can be deduced, the correlation coefficient between the time that has passed, and the number of global confirmed cases is strongly positive. This implies that there exists a strong positive correlation between these two variables, which of course, is quite concerning for our society presently. We can plot the clusters of data to generally see this trend.

```
plot(X_global, Y_global, 'ro') % plot correlated features
xlabel('Date Numeric Translation') % add x label
ylabel('Confirmed Global Cases') % add y label
title('COVID-19 Confirmed Global Case Trend, as of April 28th 2020')
```



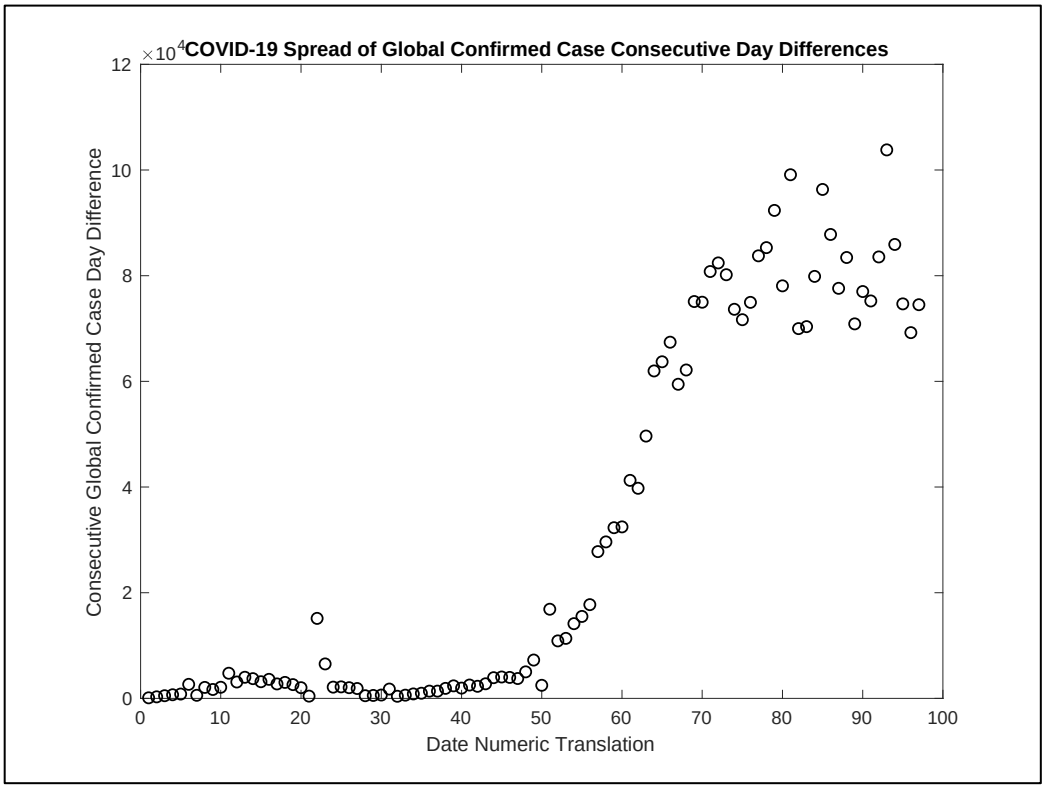
Central Tendency Measures

The question that should be asked now is whether or not the global efforts to help reduce the spread of the pandemic have been effective or not. To evaluate this, we should look at the running difference between consecutive numbers of confirmed global cases, and evaluate their spread, and range. Standard deviation, mean, and median are excellent measures of spread, and the range can be calculated using max() and min() MATLAB built-in functions. First, we need to create this new matrix using the diff() function, then we can calculate these values, and plot the spread of the data as follows:

```
% Central Tendency Measure Analysis

new_global = dataset_machine_global; % assign new global dataset
new_global = diff(new_global(2,:)); % create difference vector

centraltend = [mean(new_global) std(new_global) median(new_global)]
max_global = max(new_global)
min_global = min(new_global)

plot(new_global, 'ko') % plot spread of new global
title('COVID-19 Spread of Global Confirmed Case Consecutive Day Differences')
xlabel('Date Numeric Translation')
ylabel('Consecutive Global Confirmed Case Day Difference')
```

```
centraltend =

   1.0e+04 *

    3.2103    3.5504    0.7253
```

```
max_global =

      103789

min_global =

      99
```



From the central tendency analysis, we can see that the mean is a whopping 32,000 cases with a median of 35,500 cases and a standard deviation of 7200 cases. This implies that most of the days that have passed by have entailed a major daily increase in the number of global COVID-19 cases. Also, we can see from the spread of the data that there aren't any major data sinks, i.e. the daily number of coronavirus cases has only been increasing, or in certain cases approximately constant. It is evidently constant in the beginning (up to 50 days from January 22nd, 2020), but then blows up afterwards. This could perhaps have to do with the number of people who began to be tested worldwide. In other words, perhaps it was only around 55 days after the pandemic had started that countries worldwide began testing people for the virus. This is certainly concerning as it shows the poor management of COVID-19 testing worldwide. The large standard deviation is also concerning as it points out that the increasing number of cases may continue to blow up for an extended period of time in the future. Perhaps from another angle, by looking at the plot, though, despite several fluctuations on the right-hand side, it seems that this increase may be flattening rather than blowing up. Even though it's flattening at a rather high value, which is concerning for us all, it's still perhaps an indicator of the global effort to stop the spread of the pandemic; it is certainly better to have a constant increase in the number of cases, rather than a constantly increasing increase in the number of cases! Of course, a negative trend in the Coronavirus case increase would be the most favorable, but it seems like that it is too soon for that, unfortunately.

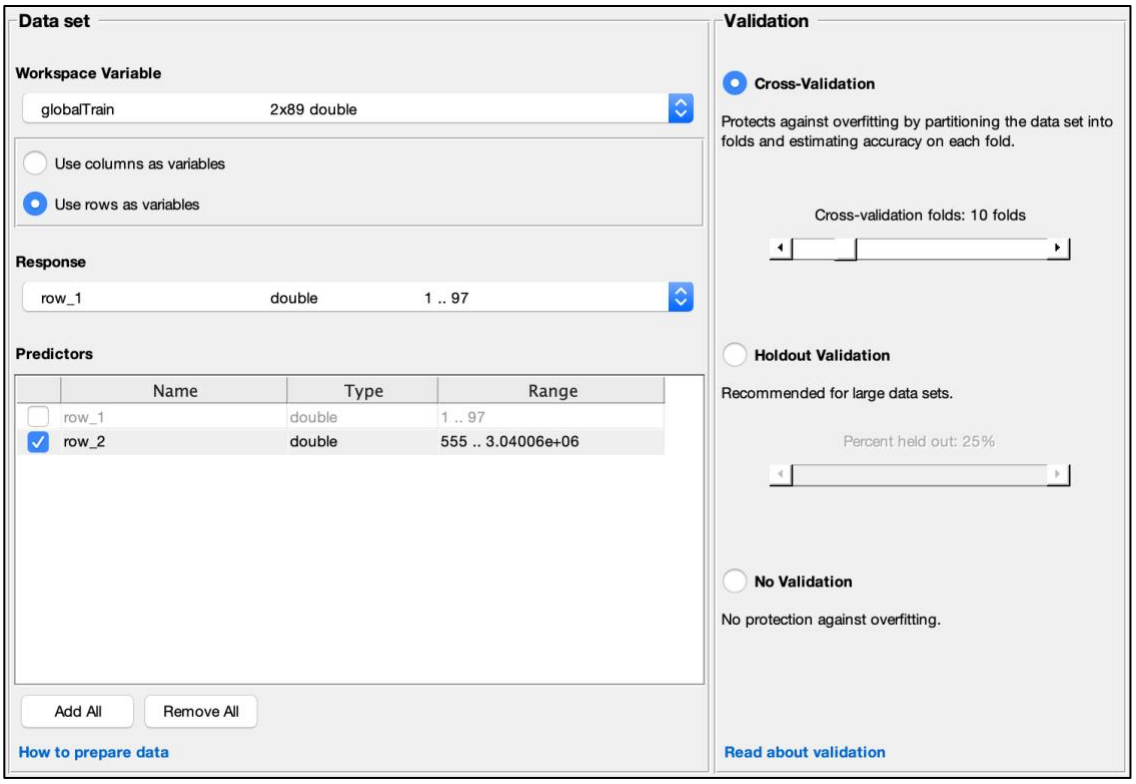
Machine Learning Toolbox (Regression Learner)

After having scrubbed our dataset, and analyzed its statistical features, we have concluded that the outbreak will not be immediately ending anytime soon due to the unfavorable trends found. As a result, it certainly begs the question: well, then, when will the outbreak end? In order to make accurate predictions, we have to use Machine Learning algorithms. Luckily for us, MATLAB has a built-in Toolbox that we will use, rather than code our own libraries and algorithms! For this particular scenario, we are interested in the regression learner, which uses predictors to provide a real number response. In order to create a model, we have to use a training dataset However, after we create the model, we need to somehow test it, thus the training dataset will not be the entire dataset. We will use a 90-10 split, which means 90% of the dataset will go into training the model, and 10% of the dataset will be used to test the model. After we have trained and tested the model, we can use it to make predictions, and in this case, predict the end of the COVID-19 outbreak!

First, we need to split our dataset into the training and testing sets. We can accomplish this using the cvpartition() function, which randomly partitions the data; we wouldn't want a manual partition as that could lead to biased training-testing splits! We do this by 'holding out' the data. This is given as found below:

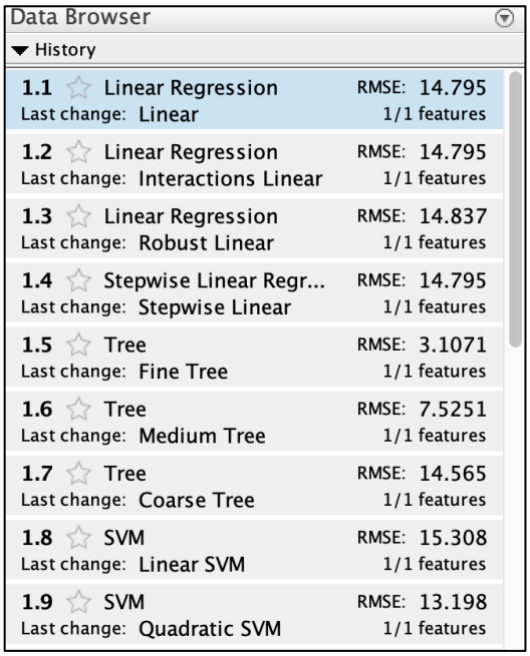
```
% Cross Validation (Train: 90%, Test: 10%)

cv = cvpartition(size(dataset_machine_global,2), 'HoldOut', 0.1);
idx = cv.test;

% Separate to training and test data
globalTrain = dataset_machine_global(:, ~idx);
globalTest = dataset_machine_global(:, idx);
```

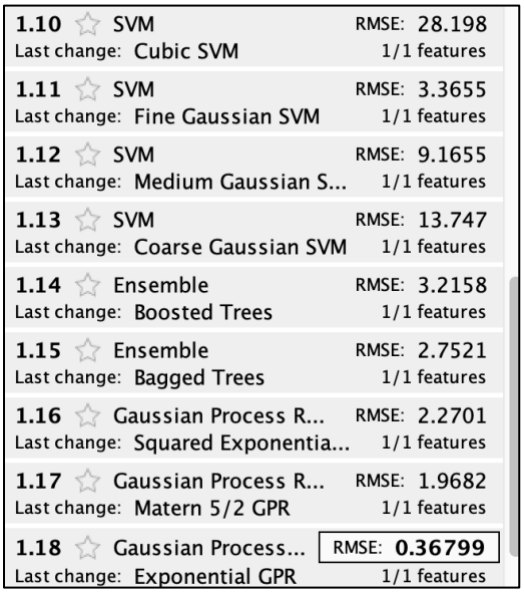
After splitting up the data, we can now train our model using globalTrain dataset. We click on the APPS tab in MATLAB, then click on Regression Learner. We use the second row as the predictor row, since we want to predict the date using the number of confirmed cases, after all; thus, the first row (the numeric data array) represents the model's response. We pick at least 10-folds for the cross-validation. This is because having this recommended number of folds can prevent overfitting the data, and leading to inaccurate predictions. It prevents overfitting by partitioning the data into the number of folds selected, then creating a model for each fold and estimating its accuracy.



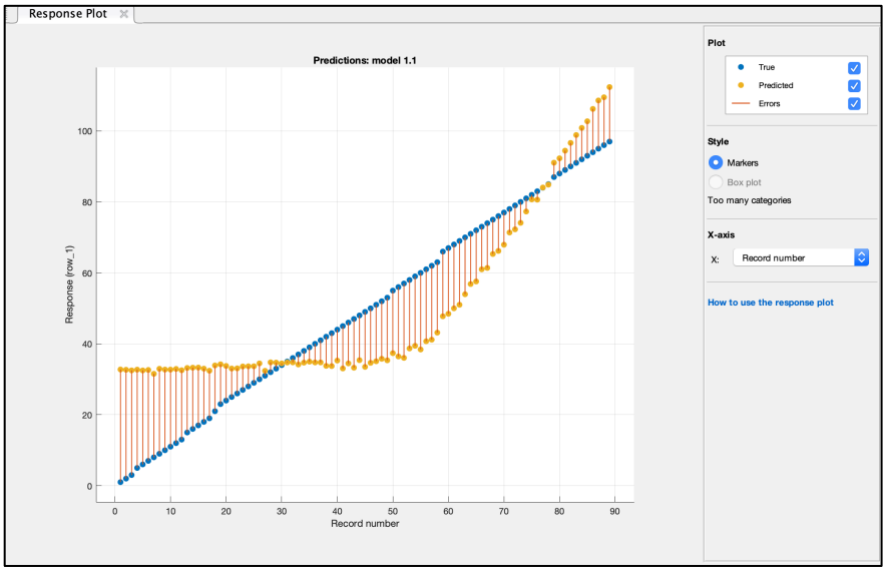
Then, we press Start Session, and train the model by running all the possible regression learning algorithms by clicking 'Train all Model types.' We do this to compare the different accuracy rates of the algorithms used and pick the one with the optimal accuracy rate; determined by the root-mean squared deviation (RMSE).

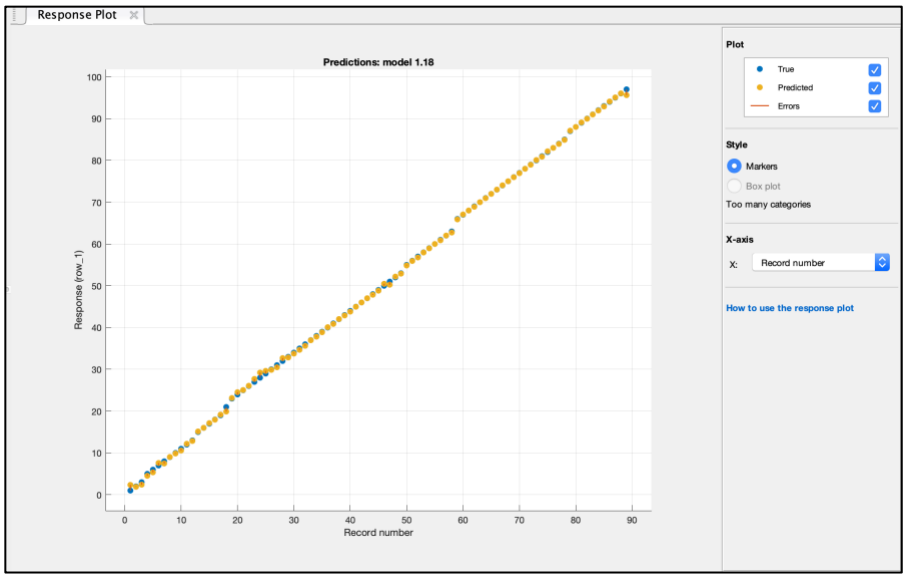
After training the model, we get the following algorithm results:





From the algorithm history, we can see that the linear regression algorithms produced the greatest error in modelling the data, thus they will not be considered. Second, the different regression tree models produced a relatively lower error, with the Fine tree producing a low error. Third, the support vector machine (SVM) models scored very high in error, except for the Fine Gaussian algorithm which scored slightly higher in error than the Fine tree model. As for the ensembles of trees models, they yield lower error than the previously mentioned models, which means they could be a good fit. Lastly, the Gaussian process regression (GPR) models yield the lowest error of all, with a whopping low of 0.36799 RMSE for the exponential GPR. This is not surprising as the trend does somewhat exponential! We can actually view these RMSE differences graphically, by comparing, for example, the linear regression linear model to the exponential GPR model plots:





Using the Toolbox, we can evidently see that there is significantly more error (highlighted in red) for the linear regression model than the exponential GPR model; there are barely any red error bars for the latter, it's almost a perfect fit!

After having selected the model with the lowest RMSE (higher accuracy), we will export it to the MATLAB workspace and use it on the testing data set, as follows:

```
>> yfit = trainedModel.predictFcn(globalTest(2,:)) % test the model

yfit =

    3.6707
   14.0346
   20.1008
   21.0047
   53.8882
   63.8478
   64.9085
   86.0445
   96.6192
```

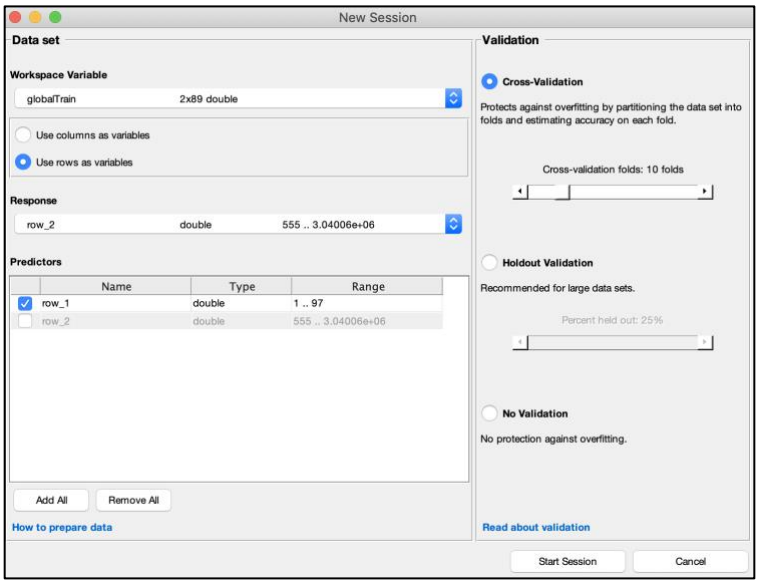
```
>> globalTest(1,:)

ans =

     4    14    20    22    54    64    65    86    98
```

Note that we have used 'Export Compact Model' to reduce the memory used in the workspace; i.e. this excludes the training dataset as we already have it in our workspace. The trained model is denoted as trainedModel in the workspace, and it is actually a 1x1 struct. We find the response of the model using the training set's predictors, then we compare the response of the model to the training set's actual response.

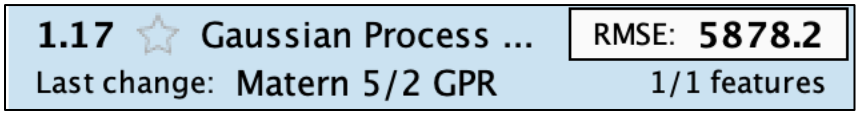
Clearly, there's very minimal error between the two sets; we can determine the error by finding the percentage difference vector and its average as found below:

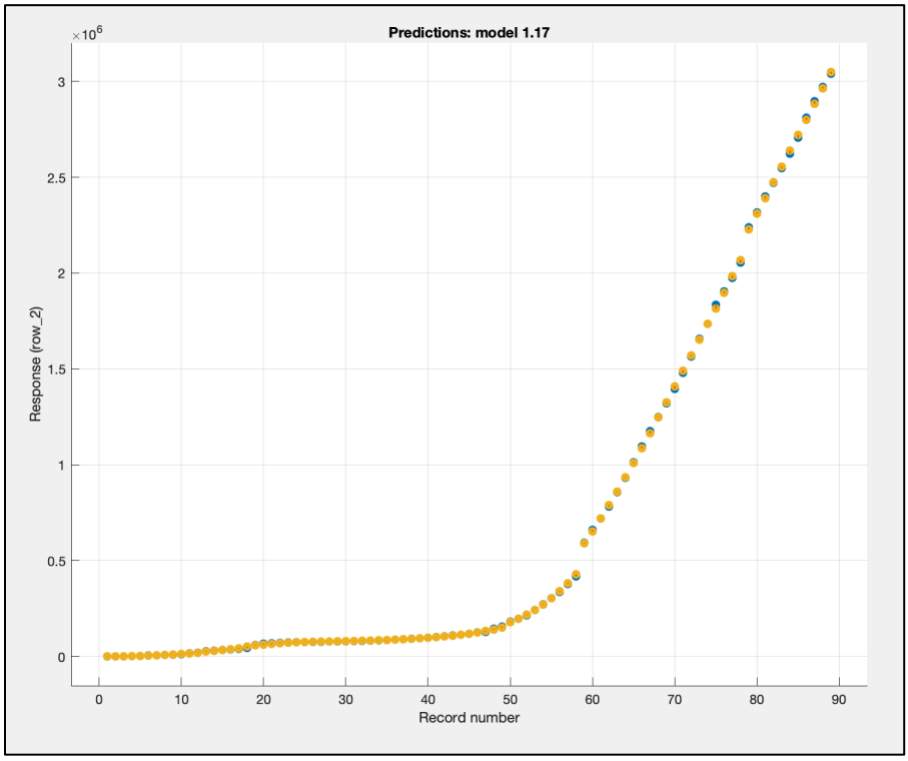
```
>> percenterror = abs((globalTest(1,:) - yfit') ./ globalTest(1,:)) .* 100

percenterror =

    8.2337    0.2475    0.5038    4.5242    0.2069    0.2379    0.1408    0.0517    1.4090

>> mean(percenterror)

ans =

    1.7284
```

Evidently, a 1.72% average percentage error isn't quite significant. Thus, now that we have verified the reliability of our trained model from its testing results, we can make predictions with regards to the COVID-19 apex date. To do this, we know that the last date entry is 98 (which represents the 28th of April, 2020), so we need to input a vector of values from 99 onwards (perhaps up to 365 days to mark a year) to determine the point at which the curve no longer increases, and remains approximately constant. However, after trying this out, it turns out that it doesn't work; this is because our predictors are the number of coronavirus cases, and the response is the numeric data array. In order to fix this, we have to create a new model in which the first row is used as the predictor, and the number of cases (row 2) is the response! We follow similar steps as above:



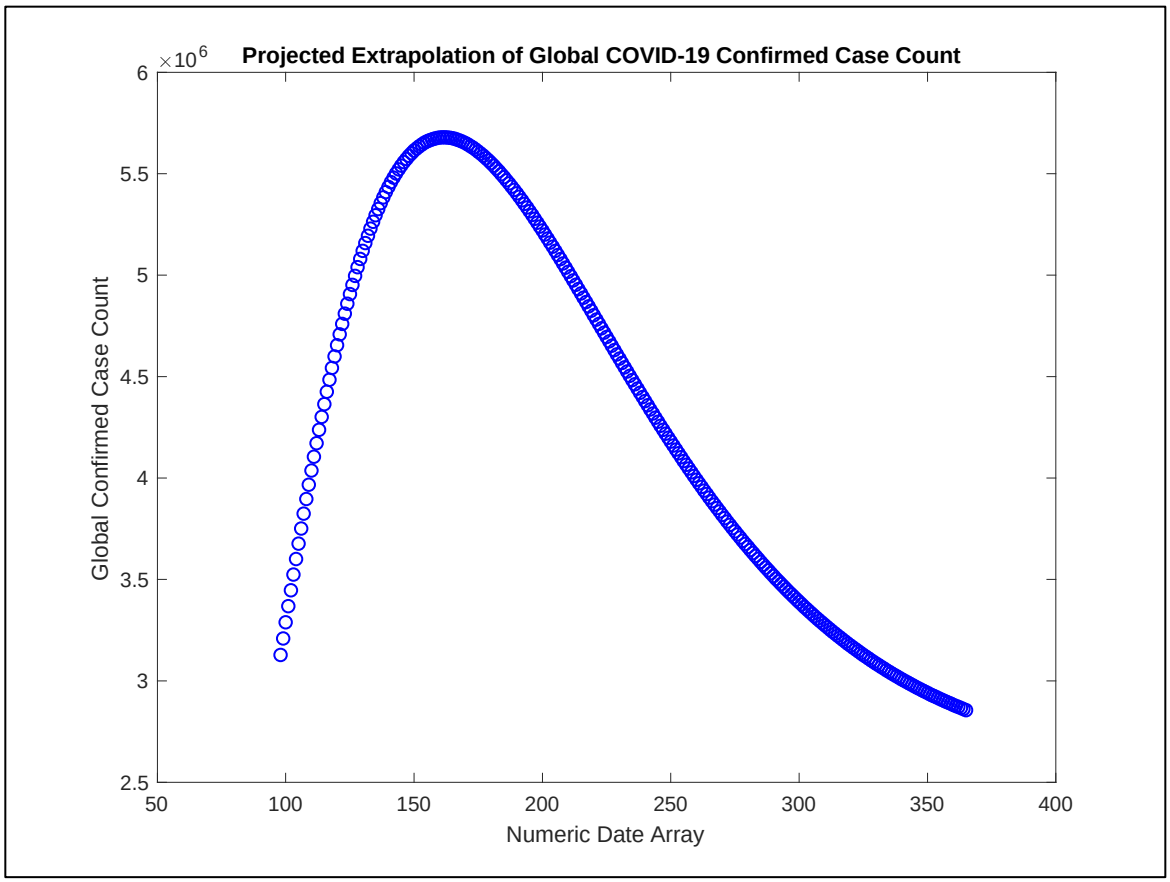
In this case, however, we get that the Matern 5/2 Gaussian process regression model is the most accurate:





This trend looks familiar! Clearly, the first regression model we used didn't look very familiar, which was a big indication that there was a problem with it. We export this new model as trainedModel into the workspace; note we already determined the accuracy from the percentage error, so there's no need to repeat this step here. Now, we input numeric data from 99 (April 29th, 2020) to 365 (around January 22nd 2021) to see when the apex is reached (if it reached within the given range!):

```
% Make Predictions using Trained Model
X_predict = 98:365;
Y_predict = trainedModel.predictFcn(X_predict);
plot(X_predict, Y_predict, 'bo')
xlabel('Numeric Date Array')
ylabel('Global Confirmed Case Count')
title('Projected Extrapolation of Global COVID-19 Confirmed Case Count')
```



We know that the number of confirmed cases cannot decrease (unless a number was overestimated somehow), so it's the maximum of the plot above that we're interested in, and the points that proceed the maximum aren't of relevant interest. From the plot, we deduce that the maximum number of global confirmed cases is predicted to occur at X = 163. If we translate 163 to a date using our conversion metric, that corresponds to July 3rd, 2020. This means that the ML algorithm says the pandemic should reach its apex by July 3rd, 2020! That's just two days after Independence Day, what a way to celebrate! This is great news as this means the pandemic would end in the beginning of Summer Session II, allowing plenty of time for universities like BU to prepare for transition back to in-person classes during the Fall semester. At the same time, it is rather depressing that the number of confirmed cases reaches 5,697,000 cases globally, implying that there will be a lot more obstacles to overcome and deaths to mourn. This means that governments should put forward even more efforts to bring down this apex number of cases by further collaboration with countries.

References

- Article Link: https://www.forbes.com/sites/cognitiveworld/2020/02/07/biased-algorithms/#63bfdc5576fc

- Big Data Set Link: https://data.humdata.org/dataset/novel-coronavirus-2019-ncov-cases