# CH5019 - Mathematical Foundations of Data Science

Jan - May 2020

## Course Project - Group **10**


ME17B140 - Naveen Reddy

ME17B153 - Miheer Athalye

ME17B158 - Omkar Nath

ME17B170 - Uma T.V.

ME17B183 - Ishan Naryani

# *Question 1*

**Introduction and Aim:**

- The dataset has 150 images consisting of 10 images of 15 subjects taken under different conditions. The goal is to compute a representative image of each subject using the method of Singular Value Decomposition. Each test image is then compared with each of the 15 representative images and the norm is calculated; which is simply the sum of the squares of all the terms of the matrix that is the difference between the test matrix and representative image. The subject whose representative image gives the least norm is considered to be the subject to which the test image is of. The accuracy rate gives us a good estimation of how accurate this method is. It measures how many images out of 150 the program is able to recognize the correct subject.

**Brief explanation on SVD and its relevance to the problem:**

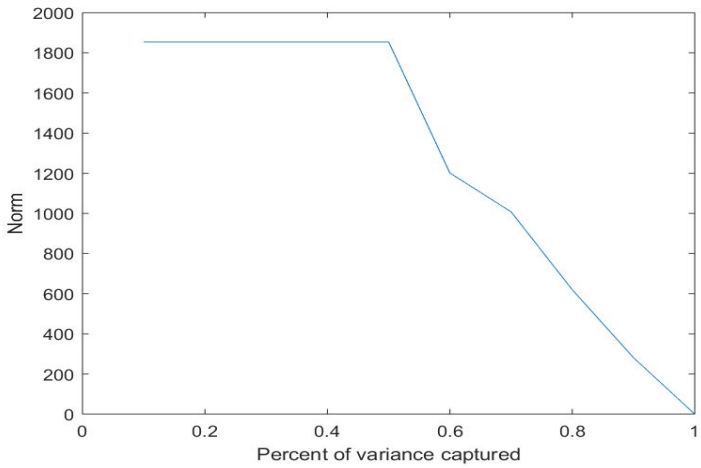
- What the SVD does is split a matrix into three important sub matrices to represent the data. Given the matrix A, where the size of a is $m \times n$ where $m$ represents the number of rows in the matrix, and $n$ represents the number of columns, A can be broken down into three sub matrices $A = U\Sigma V^T$ where $U$ is of size $m \times m$, $\Sigma$ is of size $m \times n$ and is diagonal, and $V^T$ is of size $n \times n$. U and V are the matrices with eigenvectors of $A*A^T$ and $A^T*A$ and $\Sigma$ is a diagonal matrix with square root of eigenvalues of the matrix $A*A^T$. The values of the matrix $\Sigma$ are arranged in descending order.

- It is accurate to say that $\Sigma$ captures the characteristic features in the initial values (which have higher value and hence capture more variance/information and) while the latter values are smaller and reflect the slight changes in each image of the same subject due to things like lighting, angle, pose, expressions, etc.

- Let the diagonal matrix $\Sigma$ have elements $\sigma_1, \sigma_2, \ldots \sigma_k$ such that all are greater than zero and $\sigma_1 > \sigma_2 > \ldots \sigma_k$. If the vectors of U and V be $u_1, u_2 \ldots$ and $v_1, v_2 \ldots$ of sizes m and n respectively.
  $A = \Sigma\, u * \sigma * v$ from 1 to k. The original image can now be directly constructed from U, $\Sigma$ and V.

- It is clear that higher the value of $\sigma$, the more it contributes to the image (or has more information stored in its corresponding vectors). As a result, the lesser values can be attributed to noise/features which are not characteristic of the subject and ignored. If trace($\Sigma$)=t we generally take value from $\sigma_1$ to $\sigma_r$ such that their sum is equal to a fixed percentage of t. The $\sigma$ from $\sigma_{r+1}$ to $\sigma_k$ are ignored. The higher the percentage, the more variance (which measures information) of the original image it captures (the more it becomes like the original image) and the lesser norm the new image shows from the original image.

- For example, the following is the graph of the first image of the first subject. The image is converted to matrix A on which SVD decomposition is performed. The x axis shows how much of the $\sigma$s from the resulting $\Sigma$ matrix we take while calculating the new image. The y axis measures the norm between the two images-which shows how alike they are. More the $\sigma$s we choose to take, more of the variance and features we capture and more the constructed image becomes exactly like the original image (thus reducing the norm). At 100%, we have taken all the $\sigma$s and the newly constructed image matrix is the same as the matrix on which SVD decomposition was performed-due to which the norm is zero. The flat line in the starting of the graph shows that the just the first eigenvalue of the matrix is so large that the matrix $u_1 * \sigma_1 * v_1$ is almost similar to the matrix A and captures about 60% of the variance (or the information) of the original image.




- We now have to choose how many of the diagonal elements (σs) of Σ to take very carefully. We have to strike a fine balance between not taking the basic characteristics of the subject from an image and also taking too much noise/unwanted features from the image. The representative image which will be the average image of all the 10 modified images. If we do the latter, the representative image would be muddled by extra information that is not a character of the subject's face and cause problems in facial recognition. If we do the former, we might miss some important feature of the subject face in the representative image, again causing error in facial recognition.

- The following 3 images applied to the image x1 (first image of first subject) show how one can construct an image containing just $u_1 * \sigma_1 * v_1$; $u_1 * \sigma_1 * v_1 + \ldots$ till $u_{32} * \sigma_{32} * v_{32}$ and $u_1 * \sigma_1 * v_1 + \ldots.$ till $u_{64} * \sigma_{64} * v_{64}$ which is the image itself. The first one gets the very important character of the face (the basic outline and shape of forehead, eyes, ears, etc.). The second image has most of the variance/information and shows not only all the features but also the extra noise. It's only difference from the original image is that it is a bit blurry. The third image is the original image itself.



- By multiple trial and errors-we have chosen to take σs such that $\sigma_1 + \sigma_2 + \ldots \ \sigma_r \geq 90\%$ trace $(\Sigma) = 0.9 *$ trace$(\Sigma)$.

**Programs:**

- With the report are attached files f.m,l.m,d.m,code.m and dataq1.m which include all the programs and data for this question.
- The dataq1.m file contains all the 150 matrices corresponding to the 150 images provided. They are given names x1,x2..x150.
- The function f.m performs SVD on an image matrix. It takes in the image matrix as input and outputs the matrix which is supposedly successful in capturing all the important characteristic features of the face of the subject while ignoring the noise and unwanted details.
- The function l.m takes input as all the images of the same subject, gives it to the function f.m and then takes the output. After that it simply finds the average of these matrices and assigns the output matrix to be the representative image The representative images are stored as i1 to i15 and are shown below.

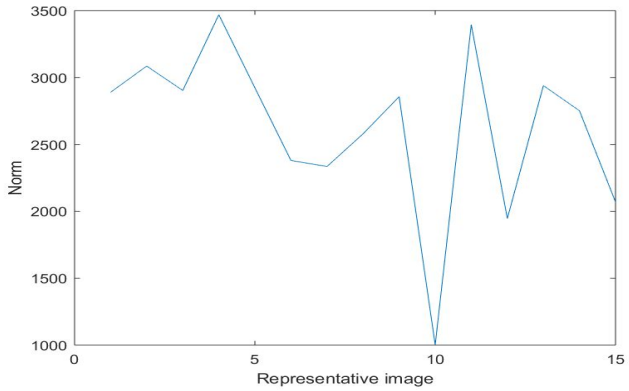
- These are the 15 representative images of the 15 subjects in the same order as was provided in the question. Some image sets of subjects have a lot of noise in them in terms of lighting,pose, angle etc.. This noise needs to be eliminated in order to get the basic and important features. Some subjects showed a lot of variation in their images while some subjects had all their images pretty much the same. In averaging, if the images are quite different- the resultant average appeared blurry. As a result some representative images show the subject clearly while others appear hazy. However all of them have the important, characteristic features of the subject that make their face unique and identifiable and have less of extra,unnecessary information.

- The function d.m simply takes input as two image matrices as input and gives the norm between them as the output. Lesser the norm, more similar the images are to one another.

- The code.m file has all the code which can simply be copied and executed directly. The code has been explained by the comments. In short, it forms a matrix x from all the x1 to x150. All the matrices of each subject are given to the functions f and l. Each image of the subject is thus processed and the representative image of the subject is formed from the average of these processed images. These representative images are stored as i1, i2 to i15.

- Now is the time for testing. A matrix h stores the norms of every image out of the 150 with respect to each representative image. Naturally its size is 150 × 15. The vector p (of length 150) now finds the index at which the norm is minimum for each row. One row represents one test image out of 150 while the column with minimum norm is the representative image closest to the test image. The index of the column is the subject to which the test image belongs. Thus p is a vector which assigns each of the 150 images to a subject. To find the accuracy, a vector t is created which has all the test images assigned to the correct subject. The accuracy rate is simply how many values in p match that in t. After some calculation it turns out to be 99% which is quite high. Thus our program is able to identify most of the images correctly.

- The image below shows the 96$^{th}$ row of matrix h plotted. It shows that the 96$^{th}$ image when compared with the 15 representative images has the least norm at the 10$^{th}$ representative image. Thus we can conclude that this image is of the 10$^{th}$ subject.




**Results and conclusion:**

- The accuracy rate is now 99% (or the error is 1%). This means that the method and program is fairly robust in facial recognition. We did succeed in making a successful facial recognition algorithm. The accuracy rate will vary according to how much of the diagonal elements of Σ we consider/ ignore.

**Further insights:**

- Each representative image matrix has resolution 64 × 64. MATLAB could easily calculate the norm for this case. However at higher resolution it can be computationally expensive to find the norm between a test image and this. Then, we can use a method called Principal Component Analysis (PCA).

- PCA simply diagonalizes a matrix-writes it as a product of 3 matrices. The first and last one consist of all the eigenvectors and are orthogonal while the second matrix is a diagonal matrix consisting of all the eigenvalues which are arranged in descending order. It can be proven that most of the information/variance of a data lies along the eigenvectors and is proportional to the eigenvalue. Thus, we can reduce the dimensions of some data without losing much information if we simply change the axis of measurement to that of the eigenvectors corresponding to the largest eigenvalues.

- We now turn the image matrix to a vector, multiply it with the transpose to form a covariance matrix and apply PCA. If we want only k dimensions out of x dimensional data-we simply take only the first k eigenvalues of the diagonal matrix and the first k eigenvectors of the other two matrices. The product of these matrices is the matrix which has the original image vector in k dimensions. We can proceed as usual, calculate norm and identify which subject the image belongs to.

- We did try this approach to our question. While the accuracy didn't change much as it was already so high, the computational time required was indeed much lower.

# *Question 2*

**Note:** Kindly refer to the Python Code attached for Question 2. All the formulations and results presented here are from this.

**Introduction and Aim:**

This data set describes operating conditions of a reactor and contains class labels about whether the reactor will operate or fail under those operating conditions. Our job is to construct a logistic regression model to predict the same.

**Statistics of the data:**

- Mean = [546.77 K, 25.49 bar, 125.03 kmol/hr, 2295.80 L/hr, 0.30 mole fraction]
- Variance = [7544.45, 203.131 1892.960, 583208.097, 0.013] units squared.
- Rank of the training dataset = 5 (full column rank)
- Ranges of the input variables are:

  Temperature: 400-700 K;
  Pressure: 1-50 bar;
  Coolant Flow Rate: 1000-3600 L/hr;
  Inlet Reactant Concentration: 0.1-0.5 mole fraction;
  Feed Flow Rate: 50-200 kmol/hr;

**Normalizing the Input data:**

The input features are scaled between 0 – 1 using the range values of the input features given. The input features are subtracted by their corresponding lowest value (e.g.: 400 for the temperature) & divided by their range (e.g.: 700 – 400 = 300 for the temperature). The scaled features are fed into the model for training and evaluation.

E.g.: If the value of the temperature feature in the input matrix is 350. Scaled value = (350-400)/(700-400) =0.5

**Hypothesis Function:**

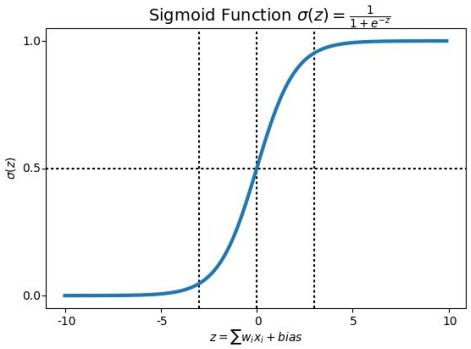
The hypothesis function used for classification is a linear model given by the following equation:

$$\mathbf{y = w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4 + w_5 x_5 + b = W^T.X + b}$$

where, $X = (x_1, x_2, x_3, x_4, x_5)^T$ ; $W=(w_1, w_2, w_3, w_4, w_5)^T$
W, b are weights (parameters) for the model to learn during training. **'b'** is a bias parameter.

Intuitively, it doesn't make sense for our hypothesis to take values larger than 1 or smaller than 0 when we know that our output needs to be either 0 or 1. To fix this, we change the form for our hypotheses to restrict it to the range [0,1]. Our new form uses the "Sigmoid Function," also called the "Logistic Function"

$$f(x) = \frac{1}{1+e^{-(x)}}$$




The function shown here, maps any real number to the (0, 1) interval, making it useful for transforming an arbitrary-valued function into a function better suited for classification. Our hypothesis will now give us the **probability** that our output is 1. Our probability that our prediction is 0 is just the complement of our probability that it is 1 (e.g. if probability that it is 1 is 70%, then the probability that it is 0 is 30%).

**Loss Function:**

The loss function used is cross-entropy loss function which is formulated as follows:

$$L(W, b) = -(1/N) \sum (y_i \,.\, \log(y(X_i)) + (1 - y_i) \,.\, \log(1 - y(X_i)))$$

(the summation is from i = 1 to N (total number of training samples or batch size)

Here $y(X_i)$ is the output value of the hypothesis function for the given input $X_i$ and $y_i$ is the target label for the given input $X_i$

If our correct answer 'y' is 0, then the cost function will be 0 if our hypothesis function also outputs 0. If our hypothesis approaches 1, then the cost function will approach infinity. If our correct answer 'y' is 1, then the cost function will be 0 if our hypothesis function outputs 1. If our hypothesis approaches 0, then the cost function will approach infinity.

Note that writing the cost function in this way guarantees that J(θ) is convex for logistic regression

**Initializing Weights:**

The input data matrix is full column rank. When the training dataset is full rank and the training model is linear the cross-entropy loss function is convex. So, there is a unique minimum where the loss function is minimum. So independent of the initialization point the function will converge to the local minima with increased training.

**Note:** Different initialization points were tried & the loss function converged for all the points tried. For simplicity we used zero Initialization.

$$W = (0, 0, 0, 0, 0)^T \;;\; b = 0$$

**Training**

The given dataset is randomly split into training & test data in 7:3 ratio respectively. The model is trained using the gradient descent algorithm using the cross-entropy loss function as the objective function for optimization. After training the model for 1000 iterations 95.67 % accuracy is achieved on the testing data and 94.00 % on the training data. The testing accuracy > training accuracy, which clearly shows that the model does not overfit.

The values of W, b obtained at the end of 1000 iterations are as follows:

**W = (-0.36141331, -0.68820591, -0.81861798, 5.05891771, -0.29389473)$^T$**
**b= -0.8777059229807475**

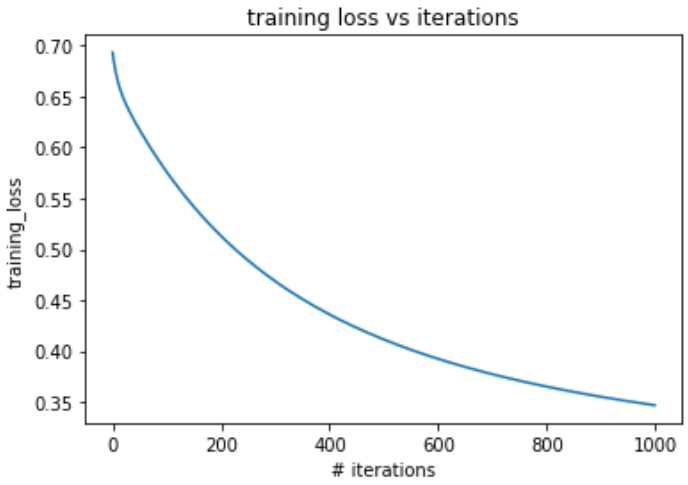
**Confusion Matrix:**

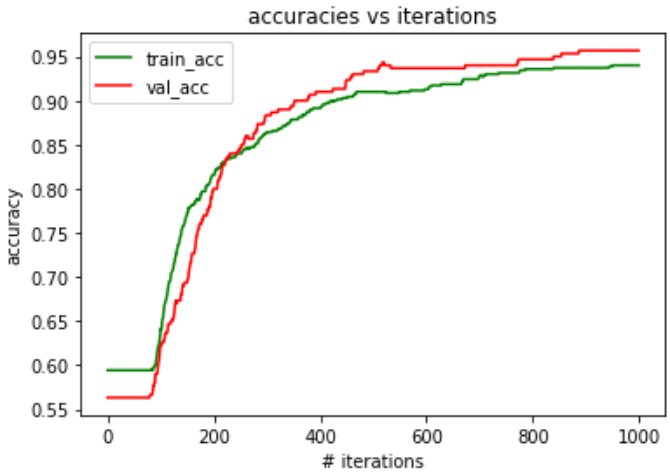
| | | Predicted | |
|---|---|---|---|
| | | Pass | Fail |
| True Labels | Pass | 162 | 7 |
| | Fail | 6 | 125 |

**Precision = TP / (TP+FP) = 0.9642857142857143**
**Recall = TP / (TP+FN) = 0.9585798816568047**

**F1 Score = 2.(precision . Recall) / (precision + Recall) = 0.9614243323442135**

**Training Metrics:**




**Further Insight:**

The support vector machine is another model used to solve classification problems. The algorithm creates a hyperplane or line(decision boundary) which separates data into classes. SVM tries to finds the "best" margin (distance between the line and the support vectors) that separates the classes and this reduces the risk of error on the data. Some differences between the two algorithm are as follows:

1. SVM works well with unstructured and semi-structured data like text and images while logistic regression works with already identified independent variables. For the given problem, we have the latter. Thus, logistic regression works extremely well.
2. The risk of overfitting is less in SVM, while Logistic regression is vulnerable to overfitting. Since we have a large amount of data(in comparison to the number of features), we need not worry about overfitting. Logistic regression gives us extremely good results for the given problem.

**Additional Notes:**

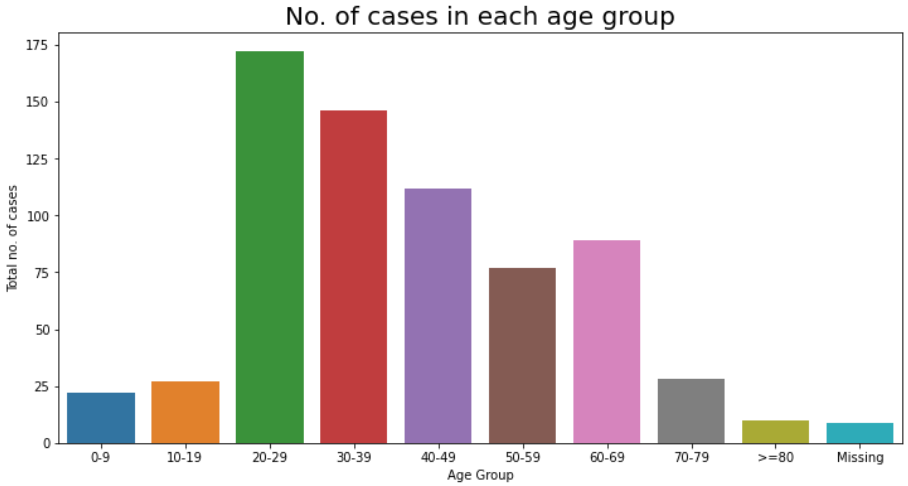
We also tried another method for the implementation of logistic regression assuming a complex model(using polynomial terms in addition to the linear terms) and using a regularisation parameter to prevent overfitting. This was done in matlab and the code has been attached. However, this is an overkill for the given problem and all the metrics and graphs we have shown in this section are from our python code.

# *Question 3*

**Note:** All the datasets used in the report are from the files uploaded on Kaggle as on 5th May 2020.

**1. Which age group is the most infected?**

The age group of 20-29 years is the most infected by COVID19. (24.86%)

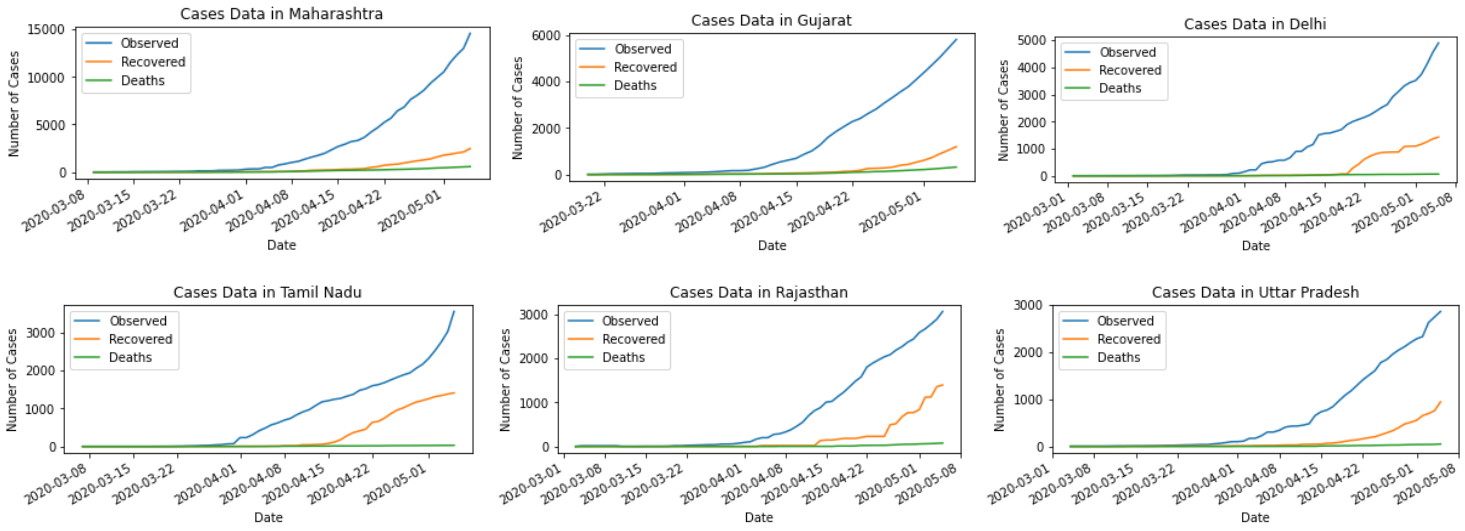


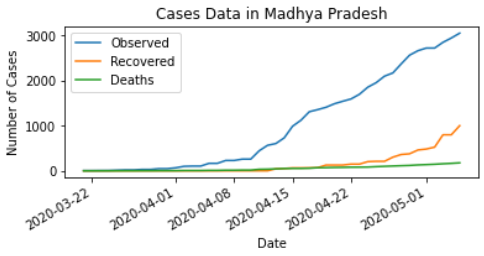

Approach:
AgeGroupDetails.csv contains the number of cases and the percentage of cases for age groups of range 10. A bar graph is plotted with the age groups in the X axis and the corresponding number of cases for the age group in the Y axis.

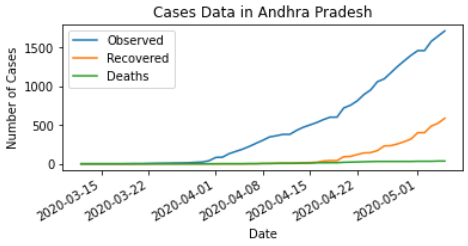
**2. Plot graphs of the cases observed, recovered, deaths per day country-wise and state wise.**

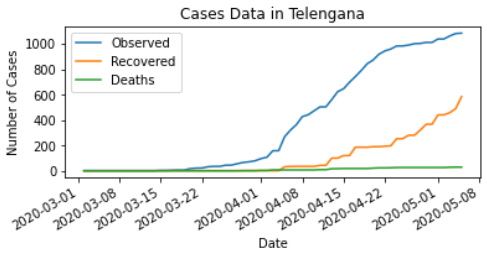
A graph for every state that shows the number of cases observed (blue plot), number of cases recovered (orange plot) and the number of deaths (green plot) in that state plotted against time (date).

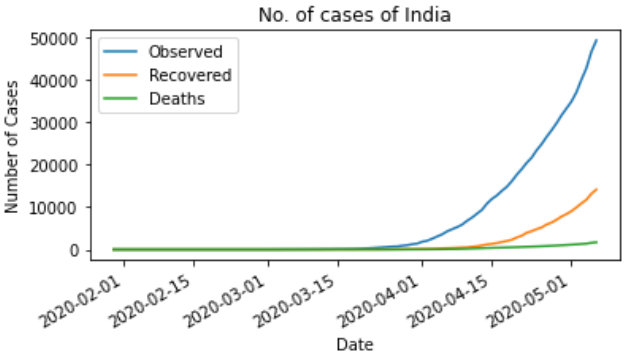
Approach: The data from covid_19_India is used to obtain the required plots using data manipulation on pandas dataframe. Seaborn is used for data visualization.





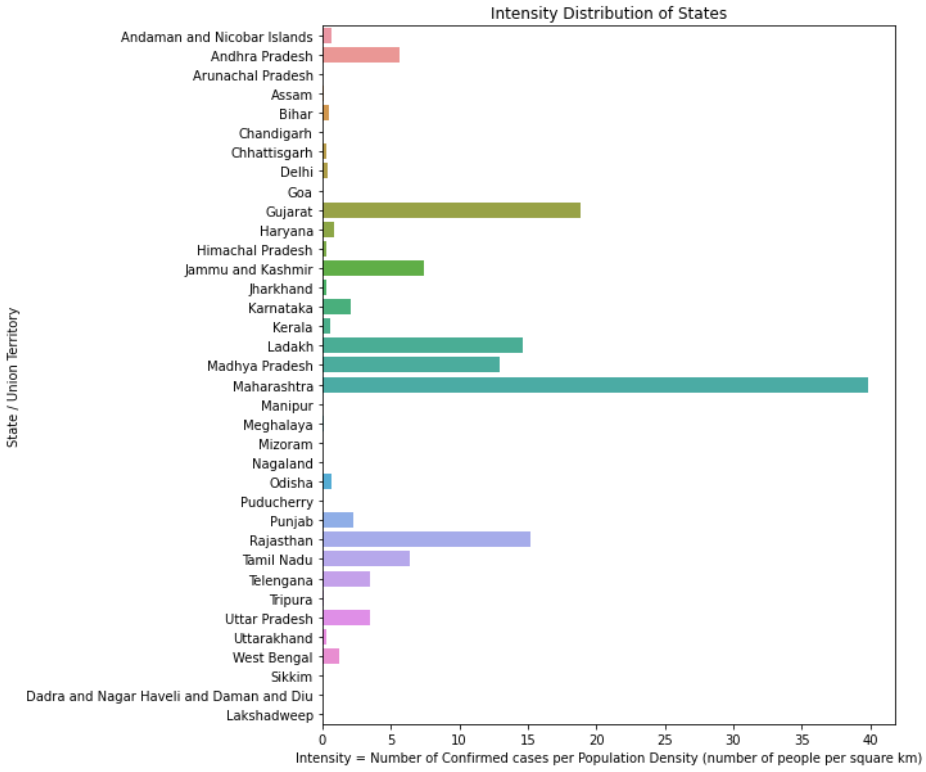




For graphs of the remaining states, kindly refer to the code.

**3. Identify the positive cases on a state level. Quantify the intensity of virus spread for each state.* Intensity here means number of positive cases/population density.**

The graph of the intensity in the units of number of confirmed cases per population density in number of people per square km is plotted. From the graph, we conclude that Maharashtra has the maximum Intensity of Virus spread.




Approach:

The positive cases on a state level can be identified with data manipulation on a dataframe created with covid_19_india.csv. The intensity of virus spread for every state is found from merging the DataFrames covid_19_india and population (from population_india_census_2011) after performing the required data manipulation.

**4. List places in the country which are active hotspots/clusters as on 10.04.2020.* Hotspot is defined as an area in a city where 10 or more people have been tested positive.**

Assumption**:** Hotspot is defined to be a district.

Approach:

For this, we use data from IndividualDetails.csv. The number of active cases is computed district-wise and totalled as of 10th April 2020. Those with 10 or more cases are defined as hotspots. The exception is Delhi which is treated as a state as no district data is available for it. The following is the list of all Districts which are Hotspots as of on 10th April 2020.

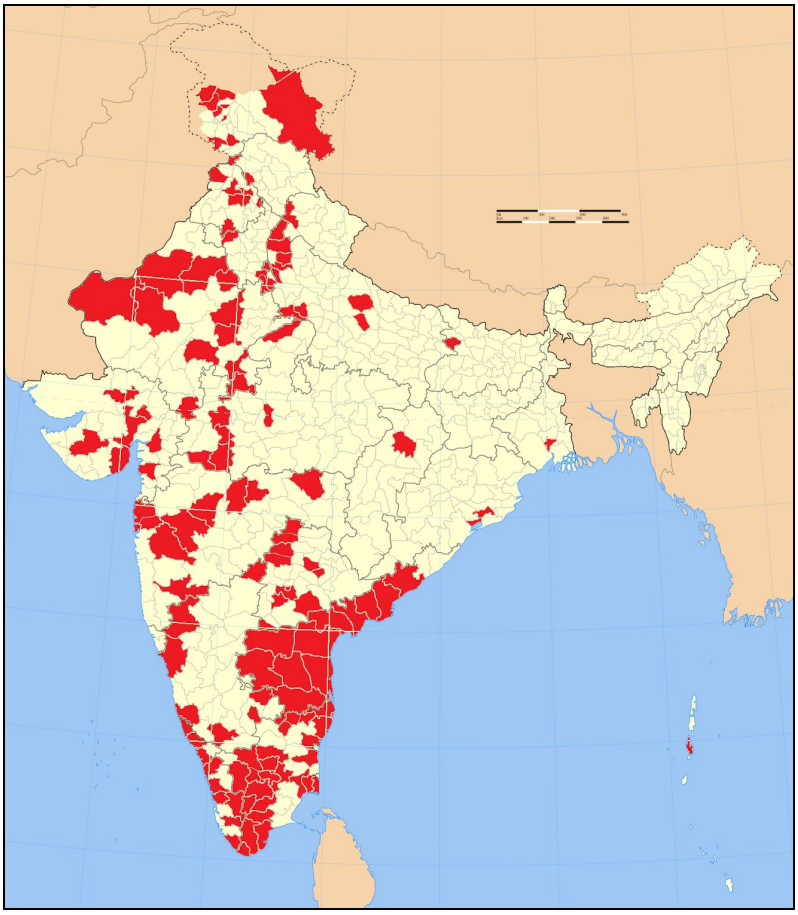
Districts:

| | State | District | Cases |
|---|---|---|---|
| 1 | Maharashtra | Mumbai | 1006 |
| 2 | Delhi | NaN | 899 |
| 3 | Maharashtra | Pune | 238 |
| 4 | Madhya Pradesh | Indore | 234 |
| 5 | Telangana | Hyderabad | 223 |
| 6 | Rajasthan | Jaipur | 220 |
| 7 | Gujarat | Ahmedabad | 195 |
| 8 | Tamil Nadu | Chennai | 171 |
| 9 | Kerala | Kasaragod | 159 |
| 10 | Maharashtra | Thane | 119 |
| 11 | Madhya Pradesh | Bhopal | 116 |
| 12 | Tamil Nadu | Coimbatore | 86 |
| 13 | Uttar Pradesh | Agra | 83 |
| 14 | Andhra Pradesh | Kurnool | 76 |
| 15 | Uttar Pradesh | Gautam Buddha Nagar | 62 |
| 16 | Gujarat | Vadodara | 59 |
| 17 | Tamil Nadu | Tirunelveli | 58 |
| 18 | Andhra Pradesh | Guntur | 58 |
| 19 | Tamil Nadu | Erode | 56 |
| 20 | Tamil Nadu | Dindigul | 54 |
| 21 | Telangana | Nizamabad | 49 |
| 22 | Punjab | S.A.S. Nagar | 48 |
| 23 | Andhra Pradesh | S.P.S. Nellore | 47 |
| 24 | Jammu and Kashmir | Srinagar | 46 |
| 25 | Kerala | Kannur | 44 |
| 26 | Uttar Pradesh | Meerut | 44 |
| 27 | Rajasthan | Jodhpur | 43 |
| 28 | Karnataka | Bengaluru Urban | 42 |
| 29 | Andhra Pradesh | Prakasam | 41 |
| 30 | Tamil Nadu | Theni | 41 |
| 31 | Tamil Nadu | Namakkal | 41 |
| 32 | Karnataka | Mysuru | 40 |
| 33 | Tamil Nadu | Chengalpattu | 39 |
| 34 | Odisha | Khordha | 39 |
| 35 | Haryana | Nuh | 38 |
| 36 | Jammu and Kashmir | Bandipora | 38 |
| 37 | Tamil Nadu | Tiruchirappalli | 36 |
| 38 | Tamil Nadu | Ranipet | 36 |
| 39 | Andhra Pradesh | Krishna | 35 |
| 40 | Haryana | Gurugram | 32 |
| 41 | Rajasthan | Jhunjhunu | 31 |
| 42 | Andhra Pradesh | Y.S.R. Kadapa | 29 |
| 43 | Jammu and Kashmir | Baramulla | 29 |
| 44 | Uttar Pradesh | Lucknow | 29 |
| 45 | Bihar | Siwan | 29 |
| 46 | Haryana | Palwal | 28 |
| 47 | Haryana | Faridabad | 28 |
| 48 | Rajasthan | Tonk | 27 |
| 49 | Rajasthan | Jaisalmer | 27 |
| 50 | Tamil Nadu | Viluppuram | 27 |
| 51 | Rajasthan | Bhilwara | 27 |
| 52 | Gujarat | Surat | 26 |
| 53 | Maharashtra | Sangli | 26 |
| 54 | Tamil Nadu | Tiruppur | 26 |
| 55 | Delhi | South Delhi | 25 |
| 56 | Maharashtra | Nagpur | 25 |
| 57 | Maharashtra | Ahmednagar | 25 |
| 58 | Tamil Nadu | Madurai | 24 |
| 59 | Rajasthan | Banswara | 24 |
| 60 | Uttar Pradesh | Ghaziabad | 24 |
| 61 | Tamil Nadu | Thoothukkudi | 24 |
| 62 | Telangana | Ranga Reddy | 23 |
| 63 | Andhra Pradesh | West Godavari | 22 |
| 64 | Tamil Nadu | Karur | 22 |
| 65 | Gujarat | Bhavnagar | 21 |
| 66 | Uttar Pradesh | Saharanpur | 20 |
| 67 | Rajasthan | Bikaner | 20 |
| 68 | Telangana | Warangal Urban | 20 |
| 69 | Andhra Pradesh | Chittoor | 20 |
| 70 | Jammu and Kashmir | Jammu | 19 |
| 71 | Jammu and Kashmir | Udhampur | 19 |
| 72 | Rajasthan | Kota | 19 |
| 73 | Telangana | Jogulamba Gadwal | 19 |
| 74 | Chandigarh | Chandigarh | 19 |
| 75 | Andhra Pradesh | Visakhapatnam | 19 |
| 76 | Kerala | Malappuram | 18 |
| 77 | Gujarat | Rajkot | 18 |
| 78 | Telangana | Karimnagar | 18 |
| 79 | Punjab | Shahid Bhagat Singh Nagar | 18 |
| 80 | Kerala | Ernakulam | 17 |
| 81 | Uttarakhand | Dehradun | 17 |
| 82 | Uttar Pradesh | Shamli | 17 |
| 83 | Maharashtra | Aurangabad | 17 |
| 84 | Andhra Pradesh | East Godavari | 17 |
| 85 | Tamil Nadu | Tirupathur | 16 |
| 86 | Tamil Nadu | Kanyakumari | 15 |
| 87 | Telangana | Nirmal | 15 |
| 88 | Maharashtra | Palghar | 15 |
| 89 | Andhra Pradesh | Anantapur | 15 |
| 90 | Jammu and Kashmir | Kupwara | 15 |
| 91 | Tamil Nadu | Cuddalore | 14 |
| 92 | Madhya Pradesh | Barwani | 14 |
| 93 | Gujarat | Gandhinagar | 14 |
| 94 | Madhya Pradesh | Khargone | 14 |
| 95 | Punjab | Pathankot | 14 |
| 96 | Jammu and Kashmir | Shopiyan | 14 |
| 97 | Madhya Pradesh | Ujjain | 14 |
| 98 | Gujarat | Patan | 14 |
| 99 | Tamil Nadu | Salem | 14 |
| 100 | Tamil Nadu | Thiruvallur | 13 |
| 101 | Madhya Pradesh | Morena | 13 |
| 102 | Tamil Nadu | Thiruvarur | 13 |
| 103 | Maharashtra | Buldhana | 13 |
| 104 | Tamil Nadu | Nagapattinam | 12 |
| 105 | Himachal Pradesh | Una | 12 |
| 106 | Maharashtra | Akola | 12 |
| 107 | Telangana | Nalgonda | 12 |
| 108 | Ladakh | Leh | 12 |
| 109 | Punjab | Jalandhar | 12 |
| 110 | Rajasthan | Jhalawar | 12 |
| 111 | Rajasthan | Churu | 11 |
| 112 | Jammu and Kashmir | Budgam | 11 |
| 113 | Punjab | Amritsar | 11 |
| 114 | Punjab | Mansa | 11 |
| 115 | Uttar Pradesh | Firozabad | 11 |
| 116 | Tamil Nadu | Virudhunagar | 11 |
| 117 | Tamil Nadu | Thanjavur | 11 |
| 118 | Tamil Nadu | Vellore | 11 |
| 119 | Kerala | Thrissur | 11 |
| 120 | Chhattisgarh | Korba | 10 |
| 121 | Punjab | Ludhiana | 10 |
| 122 | Uttar Pradesh | Sitapur | 10 |
| 123 | Telangana | Adilabad | 10 |
| 124 | Telangana | Kamareddy | 10 |
| 125 | Telangana | Mahabubnagar | 10 |
| 126 | Karnataka | Belagavi | 10 |
| 127 | Karnataka | Bidar | 10 |
| 128 | Karnataka | Chikkaballapura | 10 |
| 129 | Kerala | Kozhikode | 10 |

From the above data, it can be seen that there are a total of 130 Hotspots.

Of these, Mumbai has the highest number of cases.

Note that data with exceptional terms under the District column has been excluded from the above data. This includes terms such as “escapees”, “italians”, etc.



Based on the list of districts obtained, a map of India was generated. This map has all the districts of India outlined in it. Those districts that are hotspots have been shaded as Red in the map.

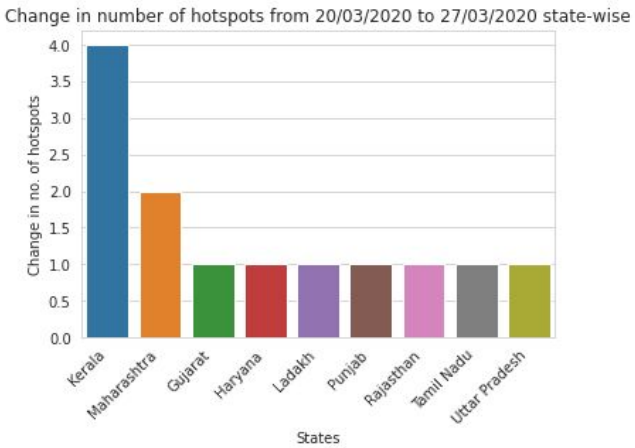
The resulting image showing hotspots as of on 10th April 2020 is as displayed here.

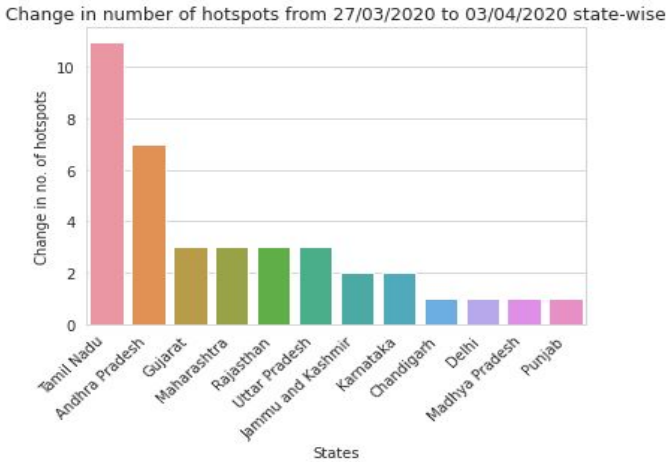
**5. Which states have the maximum change (consider increase and decrease separately) in the number of hotspots on a weekly basis from 20.03.2020 to 10.04.2020 (3 weeks).**

Assumptions:

- Hotspot is defined to be a district.
- When considering if an area is a hotspot, the number of active cases in that area at that time is considered.

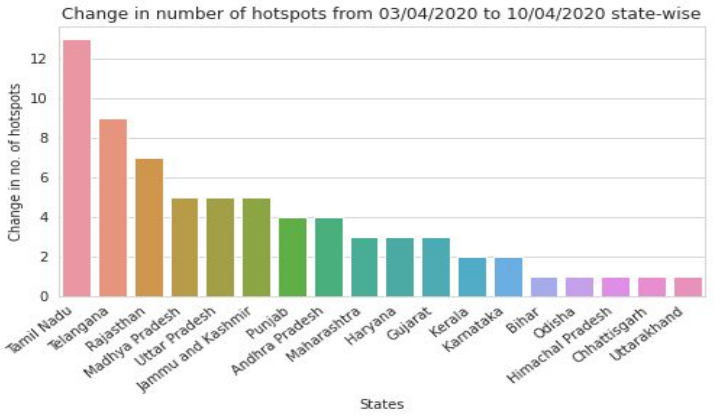
Firstly, the number of hotspots state-wise is calculated as of 20th March, 27th March, 3rd April and 10th April. From this, the change in no. of hotspots over each of the three weeks is computed by taking the difference of the above data. This is then depicted graphically in descending order of increase of hotspots as below.



The top three states with maximum increase, week wise are:
Week 1 - Kerala, Maharashtra, 7-states tie.
Week 2 - Tamil Nadu, Andhra Pradesh, 4-states tie.
Week 3 - Tamil Nadu, Telangana, Rajasthan

As for the decrease in number of cases, there is only one instance of this. Over the second week, Haryana has had a decrease of one hotspot in the state. Other than this, no other state shows a decrease in the number of active hotspots.

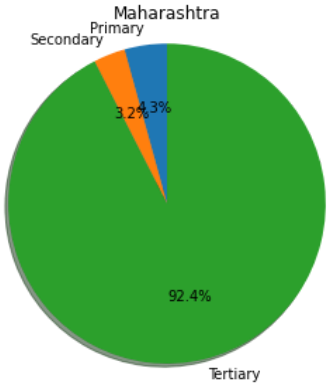


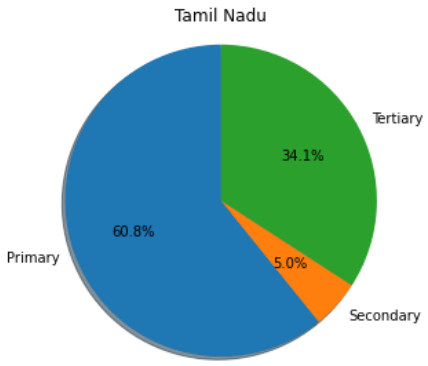

**6. For the given data, identify cases with international travel history (primary case), personal contact with primary case (secondary case). Cases which do not fall in the primary and secondary fall into tertiary cases. Quantify them based on the percentage for the top 5 states with maximum cases till 10.04.2020.***

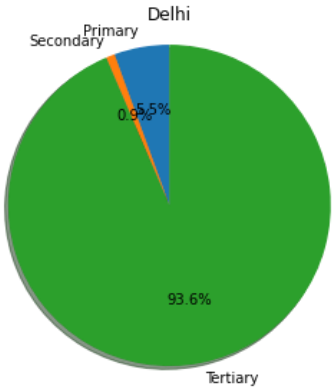
The first step is to classify the cases from IndividualDetails.csv as Primary, Secondary or Tertiary using the text data from the "notes" column of the dataframe. We used token matching using Matcher from SpaCy. The words used to match the token of text data were manually specified by going through the data and identifying segregating keywords.

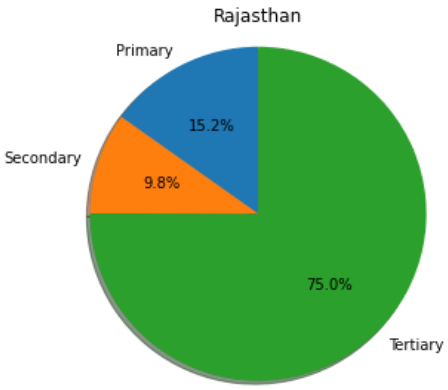
After classifying the data as Primary, Secondary and Tertiary, the top 5 states with maximum cases were quantified based on the percentage of each category, using data manipulation. The above tables show the number of confirmed cases in the top 5 states for each category, and the pie charts depicting the percentage share of the cases in every category.

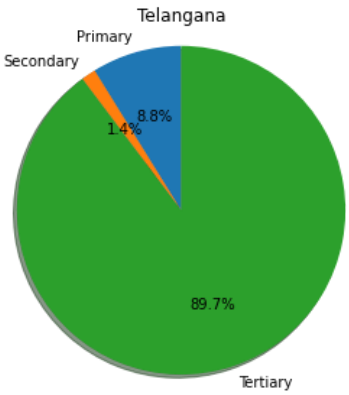
| detected_state | category | id |
|---|---|---|
| Delhi | Primary | 50 |
| | Secondary | 8 |
| | Tertiary | 845 |
| Maharashtra | Primary | 68 |
| | Secondary | 51 |
| | Tertiary | 1455 |
| Rajasthan | Primary | 85 |
| | Secondary | 55 |
| | Tertiary | 421 |
| Tamil Nadu | Primary | 554 |
| | Secondary | 46 |
| | Tertiary | 311 |
| Telangana | Primary | 43 |
| | Secondary | 7 |
| | Tertiary | 437 |



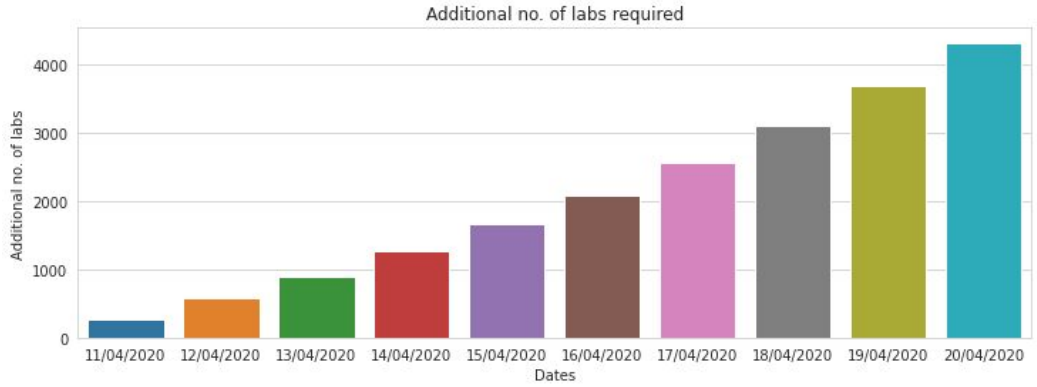



**7. Find out the number of additional labs needed from the current existing labs (assume 100 tests per day per lab) with an increase rate of 10% cases per day from 11.04.2020 - 20.04.2020. List out any further assumptions considered.**

For this, data from ICMRTestingDetails.csv is used. A correlation factor is computed between the number of positive cases detected and the number of tests that are done on a day i.e. the ratio of the two. The number of new positive cases is then computed for the required period taking a 10% increase in cases per day, using the number of cases on 10th April. The number of tests that need to be done for detecting these many cases is calculated using the correlation factor computed earlier. Finally, the required number of labs is calculated by dividing the number of tests by 100. This is done for each day in the specified period. The results are tabulated using a bar graph as below, which depicts the additional number of labs needed than the number that was used on 10th April.

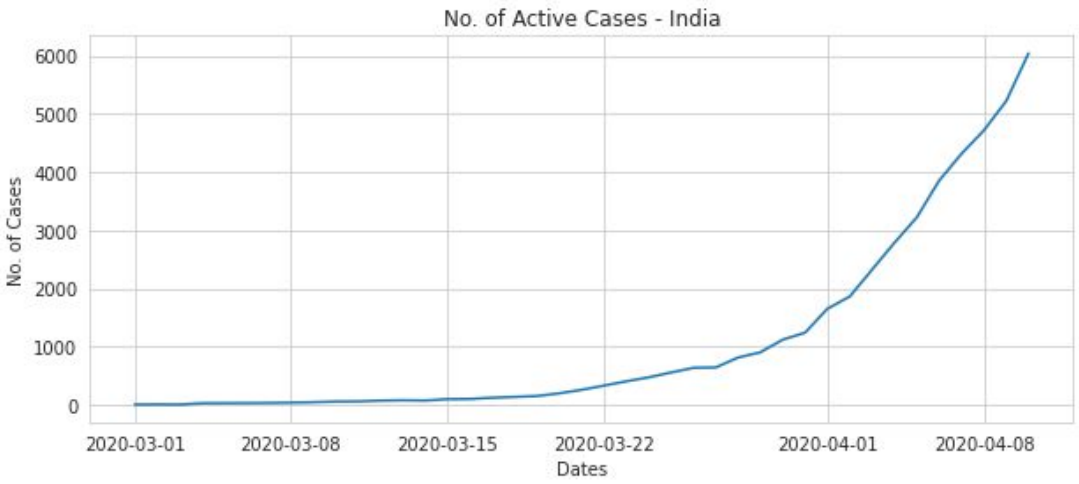



As seen, the additional number of labs needed on 20th April is 4327, for a total of 7042 labs.

**8. Plot the number of cases starting from 1st March - 10th April. Based on this plot can you comment on the popular notion of 'flattening the curve'.***

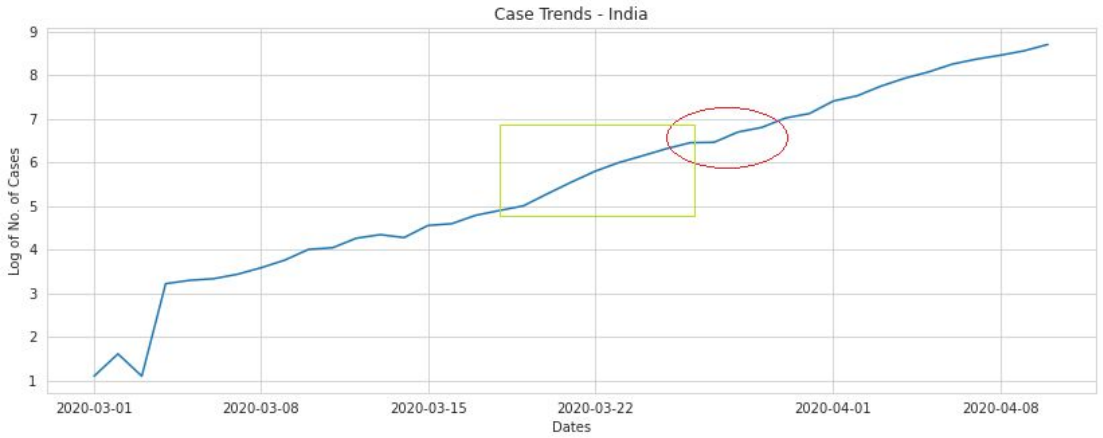
Data from the "covid_19_india.csv" file is used for this. The data is grouped by date while summing the number of confirmed, cured cases and deaths. The data is restricted to the period between 1st March and 10th April. The number of active cases is then computed by subtracting the cured cases and deaths from the confirmed cases. This is then plotted and depicted graphically.

The idea of flattening the curve is to stagger the number of new cases over a longer period, so that people have better access to care. As we see in this graph, the number of cases will keep increasing till a peak. Reducing the height of this peak means that more people will have access to medical facilities, thus saving lives. This is done by spreading out the new cases over a longer period of time.



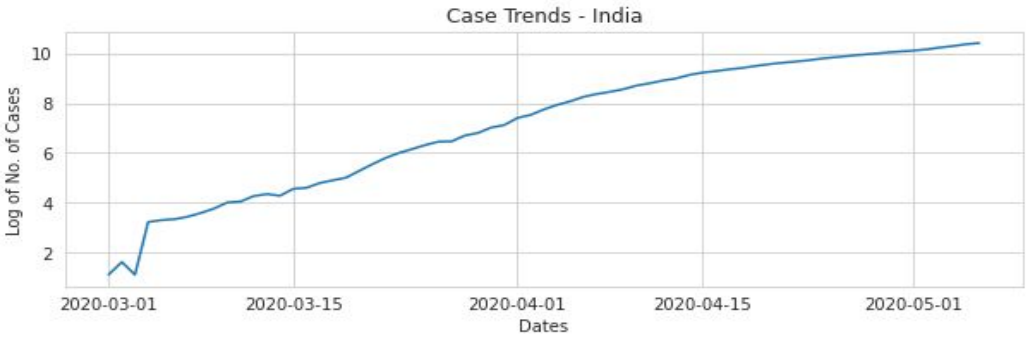
**9. As we know, social distancing is the best option to avoid the spread. Based on the time series data (covid_19_india.csv), can you suggest how successful the 21 days lockdown has been?**

Looking at the curve of part 8, it is hard to say whether the lockdown had any positive effect or not as the cases are rapidly increasing at an exponential rate. Therefore, we need to consider the graph of the log of the number of cases. The resulting graph is visible below.




As we see, in the region just before lockdown (the green box), the slope was gradually increasing causing an exponential increase in cases. Just after lockdown was declared (25th March for reference), the slope suddenly stagnates and drops (the red oval). The subsequent slope and increase is still lesser than it was previously. Although this change seems minor, it leads to an exponential decrease in the growth rate of cases. As such, we can see that lockdown did succeed in slowing down the increase of cases, working towards our objective of 'Flattening the Curve'.

Also if we look at the more recent data, we clearly see the slope decreasing which further proves the effectiveness of lockdown in reducing the growth rate of number of cases




## *References*


1. Facial recognition Technology-by Gerardus Blokdyk
2. SVD-Based Projection for Face Recognition Chou-Hao Hsu and Chaur-Chin Chen - http://www.cs.nthu.edu.tw/~cchen/ Research/2007EitFace.pdf
3. Image compression and face recognition: two image processing applications of Principal component analysis - Aleš Hladni - https://www.internationalcircle.net/international_circle/circular/issues/13_01/ICJ_06_2013_05_075.pdf
4. Section 4.3.4 of Pattern recognition & Machine Learning by Christopher M. Bishop
5. https://www.coursera.org/learn/machine-learning/supplement/AqSH6/hypothesis-representation
6. https://www.kaggle.com/sudalairajkumar/covid19-in-india