CH5019 – MATHEMATICAL FOUNDATIONS OF DATA SCIENCE

TERM PROJECT

Question1.

Every pgm file contains multiple lines. The first line contains the dimensions of the image followed by a matrix of numbers. Each image has 64x64 pixels. This means that it has 64x64 grayscale values ranging from 0-255.

Consider the following:

- Here, each image is a **realization of the sample space.**
- The 4096 values in an image can be considered as **4096 features of the image.**
- So, when there are 10 images of a class can be imagined to be **10 images that come from the same data generating process** and hence there should be some relation among them.
- Now that we have put them in a matrix the matrix can be imagined to be **the dataset**.

Successfully we have broken down this problem down to the usual **A=T+U problem** which forms an important part of discussion in the class. To identify the T and U matrices we can perform SVD, and reconstruct the matrix using only the high valued singular components.

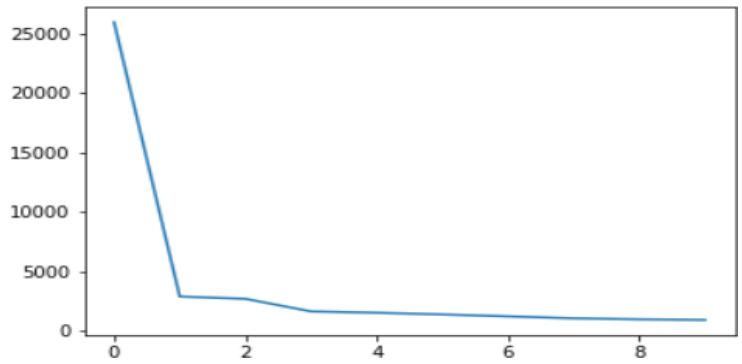
We can then use **a weighted average of the rows** (I have given equal weightage to all data points) depending on how much confidence we have on each data point and form one 1x4096 vector which serves as the representative image.

Finding the required number of components:

In order to decide a good number of components we need, instead of hard coding the number of components, I wrote a code to determine **'the elbow'** of the following singular value curve.

Fig 1a) Plot of singular values

```
[<matplotlib.lines.Line2D at 0x13f0ae27080>]
```

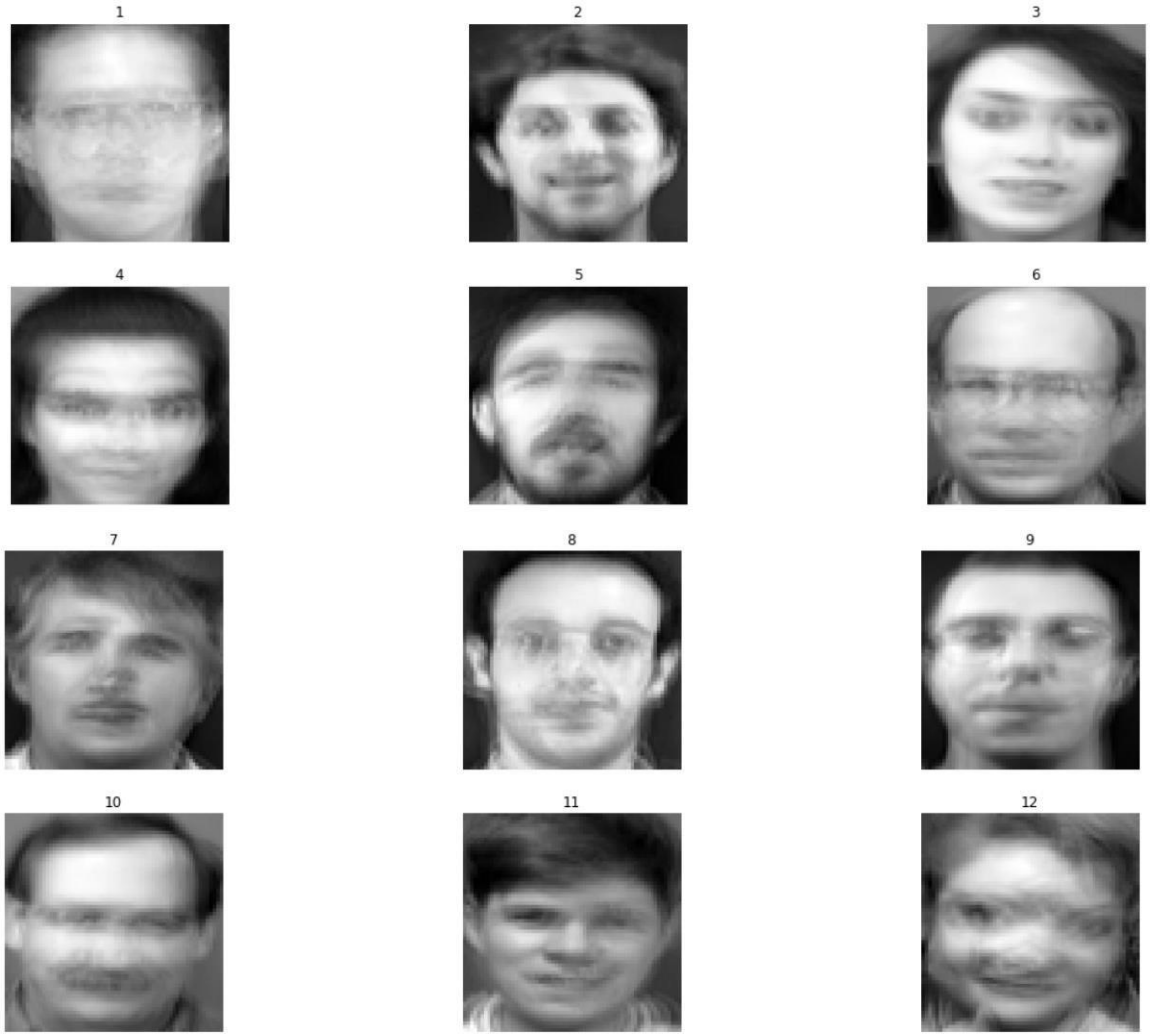


I used this equation to determine the same:

$$\sum_{i=1}^{elbow} s_i \geq 0.7 * \sum_{i=1}^{n} s_i$$

Basically, this condition means that once 70% is covered, the whole class is adequately represented.

Fig 1b) Representative images:



After finding the representative images, we classified the images based on how close they are to each representative image. The closeness was quantified by the matrix operation of 'norm'.

Results and Discussion:

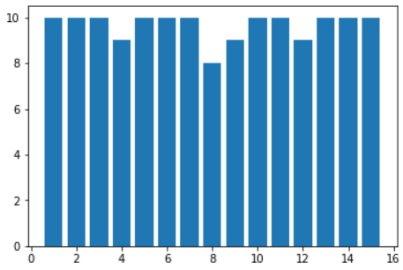
We managed to achieve an accuracy of **96.667% using L2 norm for classification** (145 correct predictions out of 150). In comparison L1 norm fetched us an accuracy of 140 correct predictions out of 150.

Fig 1c) Bar graph portraying the number of correct predictions of each class

```
In [58]: print(accuracies)
         plt.bar(np.linspace(1,15,num=15),accuracies)

         [10, 10, 10, 9, 10, 10, 10, 8, 9, 10, 10, 9, 10, 10, 10]

Out[58]: <BarContainer object of 15 artists>
```



From the graph we can infer that the **eighth class** has the least amount of correct predictions. We then, manually checked the images and realized that there was a lot of variation in the images of this class (like orientation of the face, shape of the mouth, expression, photograph illumination etc.).

This is also reflected in the representative image which is **blurred.**

Number of components used for each image:

```
array ([[3, 2, 2, 3, 3, 2, 2, 2, 2, 2, 2, 2, 3, 2, 2]])
```

We can infer that class '4' has the highest number of equally distributed components. That means it has 4 good relationships among the variables.

Question 2.

2.1 Description of the data.

Stats

	Temperature	Pressure	Feed Flow rate	Coolant Flow rate	Inlet reactant concentration
count	1000.00000	1000.000000	1000.000000	1000.000000	1000.000000
mean	546.76643	25.493270	125.029060	2295.797770	0.302692
std	86.85878	14.252407	43.508159	763.680625	0.116062
mean	1.053935e-15	2.057576e-15	5.720313e-15	3.774980e-15	-1.876055e-15
std	1.000500e+00	1.000500e+00	1.000500e+00	1.000500e+00	1.000500e+00

before normalisation

after normalisation

"Test: {Pass, Fail]}" Stats

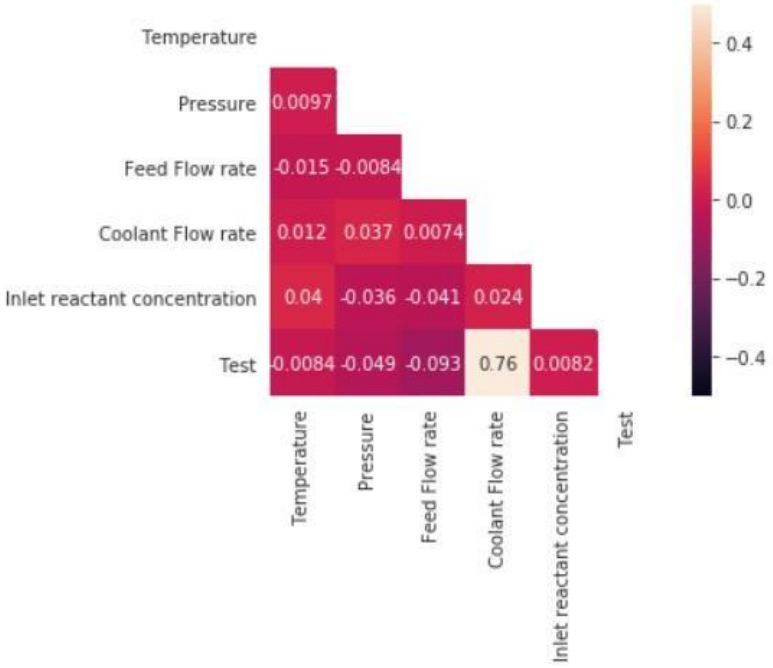
```
In [7]: data['Test'].describe()
Out[7]: count     1000
        unique       2
        top       Pass
        freq       585
        Name: Test, dtype: object
```

With the data given, we can see that mean and standard deviation of the following variables are not of the same magnitude. This will pose a problem while using gradient descent and might result in models giving unfair weightage to some features. Hence, feature scaling / normalization is done.

Test: (Pass, Fail) is converted to (1,0) respectively.

After feature scaling, the mean of the variables is evidently 0 and standard deviation being 1.

Correlation Coefficient Heat Map



2.2 Splitting the data: 700 rows are chosen randomly as training data and remaining 300 as test data. Randomness is employed to improve uniformity.

2.3 Objective Function and performing gradient descent:

The cost function for a logistic regression (with regularization) is as follows:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}\log(h_\theta(x^{(i)})) - (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2.$$

The gradient of the cost function with respect to the parameters:

$$\frac{\partial J(\theta)}{\partial \theta_j} = \left(\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}\right) + \frac{\lambda}{m}\theta_j \quad \text{for } j \geq 1$$

Multiple initializations are done to reach global minima instead of local minima. Number of iterations for gradient descent algorithm is kept at 1000 to get fair convergence with reasonably quick computing time.

We do the following initializations and there were the results obtained :

- **Rand:** Gaussian initialization of the weights(mean= 0, standard deviation = 1), Learning rate of 0.015, Regularization Parameter of 0.02

```
Iteration: 100   Logloss: 0.38189
Iteration: 200   Logloss: 0.34466
Iteration: 300   Logloss: 0.33887
Iteration: 400   Logloss: 0.33742
Iteration: 500   Logloss: 0.33706
Iteration: 600   Logloss: 0.33703
Iteration: 700   Logloss: 0.33708
Iteration: 800   Logloss: 0.33715
Iteration: 900   Logloss: 0.33722
Iteration: 1000          Logloss: 0.33727
accuracy:  0.9266666666666666
precision:  0.967948717948718
recall:  0.8988095238095238
f_score:  0.9320987654320988
Confusion Matrix
{'tp': 151, 'tn': 127, 'fp': 5, 'fn': 17}
```

- **Zeros:** Initialize all weights as 0, Learning Rate of 0.001, Regularization Parameter of 0.05

```
Iteration: 100   Logloss: 0.40755
Iteration: 200   Logloss: 0.34976
Iteration: 300   Logloss: 0.33757
Iteration: 400   Logloss: 0.33369
Iteration: 500   Logloss: 0.33207
Iteration: 600   Logloss: 0.33128
Iteration: 700   Logloss: 0.33086
Iteration: 800   Logloss: 0.33064
Iteration: 900   Logloss: 0.33051
Iteration: 1000          Logloss: 0.33045
accuracy:  0.9266666666666666
precision:  0.967948717948718
recall:  0.8988095238095238
f_score:  0.9320987654320988
Confusion Matrix
{'tp': 151, 'tn': 127, 'fp': 5, 'fn': 17}
```

- **Ones:** Initialize all weights as 1, Learning Rate of 0.001, Regularization Parameter of 0.002

```
Iteration: 100   Logloss: 0.39318
Iteration: 200   Logloss: 0.35914
Iteration: 300   Logloss: 0.35046
Iteration: 400   Logloss: 0.34777
Iteration: 500   Logloss: 0.34702
Iteration: 600   Logloss: 0.34698
Iteration: 700   Logloss: 0.34718
Iteration: 800   Logloss: 0.34746
Iteration: 900   Logloss: 0.34773
Iteration: 1000          Logloss: 0.34797
accuracy:  0.9266666666666666
precision:  0.967948717948718
recall:  0.8988095238095238
f_score:  0.9320987654320988
Confusion Matrix
{'tp': 151, 'tn': 127, 'fp': 5, 'fn': 17}
```

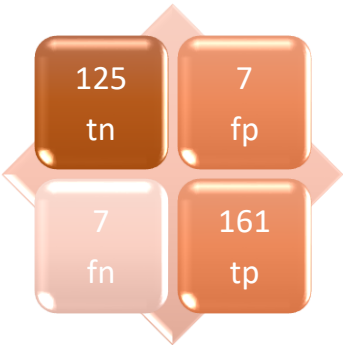
The threshold has been set to 0.4 while running the following data. We could see f1 score did not see any improvement for different initializations. The first model is again run with threshold set to 0.3. The following has been observed.

Final: Method = rand, lr = 0.0015, lambda = 0.02, threshold = 0.3

```
accuracy:  0.9533333333333334
precision:  0.9583333333333334
recall:  0.9583333333333334
f_score:  0.9583333333333334
Confusion Matrix
{'tp': 161, 'tn': 125, 'fp': 7, 'fn': 7}
```

2.3 RESULTS AND DISCUSSION

Confusion Matrix



F1 Score:

$$= 2* \frac{precion*Recall}{precision+recall}$$

$$= 2* \frac{0.95834*0.95834}{0.95834+0.95384}$$

=0.95834

Comparison with Scikitlearn's Logisitic Regression:

```
accuracy:  0.96
precision:  0.9482758620689655
recall:  0.9821428571428571
f_score:  0.9649122807017544
Confusion Matrix
{'tp': 165, 'tn': 123, 'fp': 9, 'fn': 3}
```

The F1 score of the above and our model is pretty close. Hence, our logistic regression model has converged the loss function to an acceptable minimum.

Question 3.

3.1

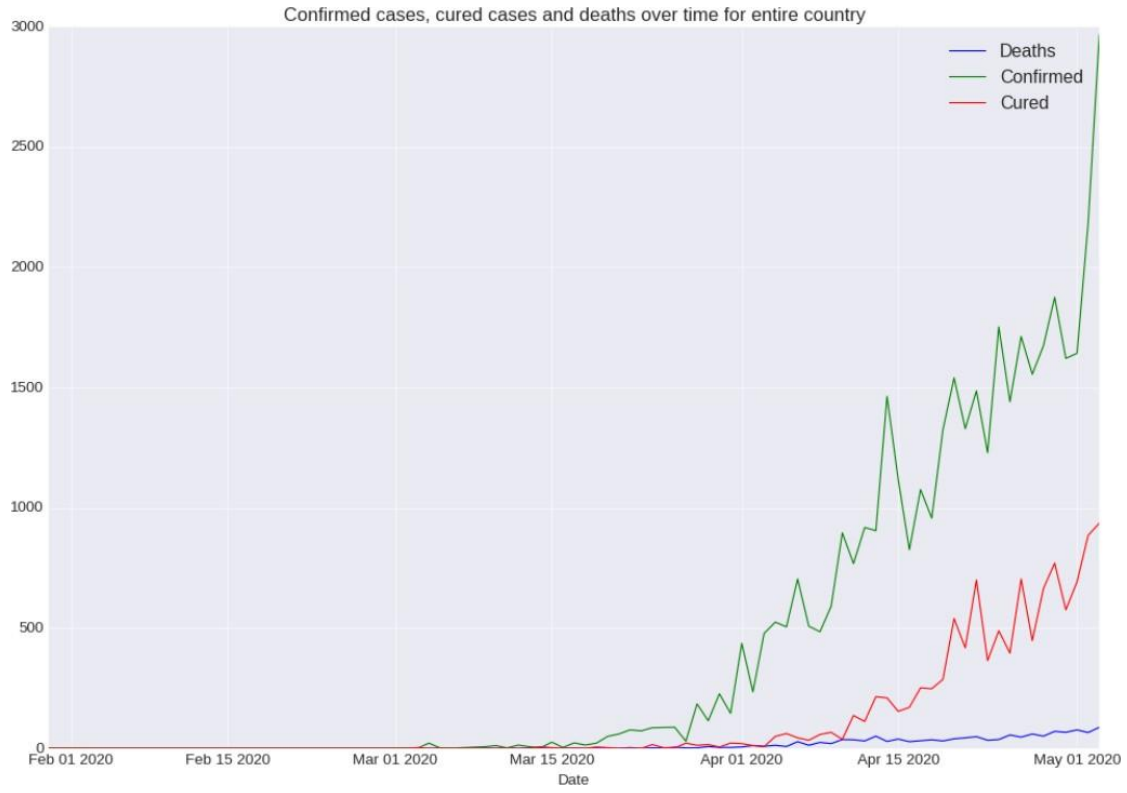
	Sno	AgeGroup	TotalCases	Percentage
0	1	0-9	22	3.18%
1	2	10-19	27	3.90%
2	3	20-29	172	24.86%
3	4	30-39	146	21.10%
4	5	40-49	112	16.18%
5	6	50-59	77	11.13%
6	7	60-69	89	12.86%
7	8	70-79	28	4.05%
8	9	>=80	10	1.45%
9	10	Missing	9	1.30%

From the Total cases column, it is evident that **age group 20-29 is most affected.** (172 cases)

3.2

Graphs have been plotted for number of cases observed, recovered and deaths. For a number of states, Negative values in confirmed cases has been observed. This is due to the infecting people moving from one state to another.

For example, an infected Bengali person in Andaman is immediately sent back to West Bengal.



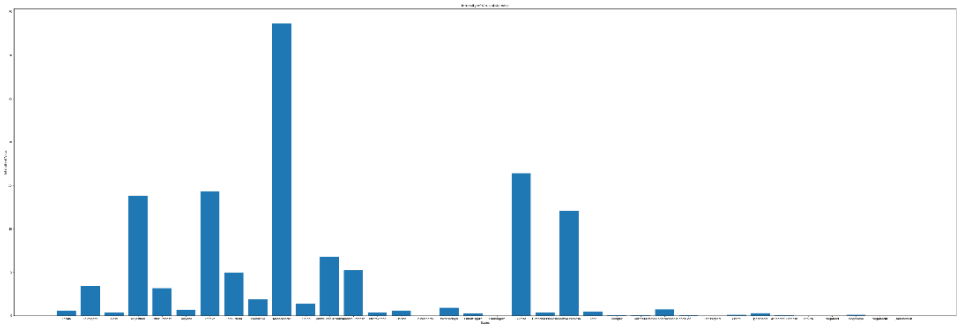
Due to the sheer number of state wise graphs, the plots are kept as an attachment for clarity.

3.3 Intensity of the virus spread is calculated as (Latest number of cases)/(Population Density). The following has been listed below as tabular column and visualization shown below.

States	Intensity
Kerala	0.582
Telangana	3.407
Delhi	0.365
Rajasthan	13.791
Uttar Pradesh	3.171
Haryana	0.688
Ladakh	14.286
Tamil Nadu	4.968
Karnataka	1.9
Maharashtra	33.688
Punjab	1.404
Jammu and Kashmir	6.796
Andhra Pradesh	5.224
Uttarakhand	0.312
Odisha	0.595
Puducherry	0.003
West Bengal	0.896
Chhattisgarh	0.228

Chandigarh	0.01
Gujarat	16.412
Himachal Pradesh	0.325
Madhya Pradesh	12.059
Bihar	0.437
Manipur	0.016
Mizoram	0.019
Andaman and Nicobar Islands	0.717
Goa	0.018
Unassigned	0
Assam	0.108
Jharkhand	0.278
Arunachal Pradesh	0.059
Tripura	0.011
Nagaland	0.0
Meghalaya	0.091
Nagaland#	0
Jharkhand#	0

Intensity of Virus spread



3.4 There were a LOT of null values in the 'detected_city' column of IndividualDetails.csv - about 6500(the size of the data-frame is itself 7600). Upon analysing the data we inferred that some of the "detected_district" names were same as that of the "detected_city", for example, Chennai is the name of the city as well as district. So we proceeded to fill the city name with the district name in such cases.

For counting hotspots, we found out how many were diagnosed on or before 10th April but their sickness status changed to recovered/deceased only after 10th April.

Active hotspots/Clusters (as of 10/04/2020):

Agra	83	Ahmedabad	188
Ahmednagar	18	Amritsar	11
Aurangabad	16	Bengaluru	21
Bhavnagar	21	Bhilwara	26
Bhopal	116	Bhubaneswar	19
Bodi	13	Chandigarh	18
Chennai	164	Coimbatore	58
Dehradun	17	Faridabad	28
Gandhinagar	14	Ghaziabad	23
Guntur	30	Gurugram	34
Hyderabad	212	Indore	234
Jaipur	202	Jammu	18
Jhunjhunu	31	Jodhpur	42
Kalyan-Dombivali	34	Karimnagar	18
Kasaragod	157	Kurnool	18
Lucknow	29	Ludhiana	10
Malappuram	16	Mettupalayam	20
Mira-Bhayandar	21	Mohali	11
Mumbai	999	Nagpur	25
Navi Mumbai	32	Nizamuddin area	24
Noida	12	Ongole	18
Palwal	28	Perundurai	17
Pimpri-Chinchwad	22	Pune	208
Rajkot	18	Ramganj	15
Sangli	26	Srinagar	42
Surat	26	Thane	27
Tirunelveli	57	Tiruppur	26
Vadodara	59	Vasai-Virar	12
Vijayawada	10	Visakhapatnam	14

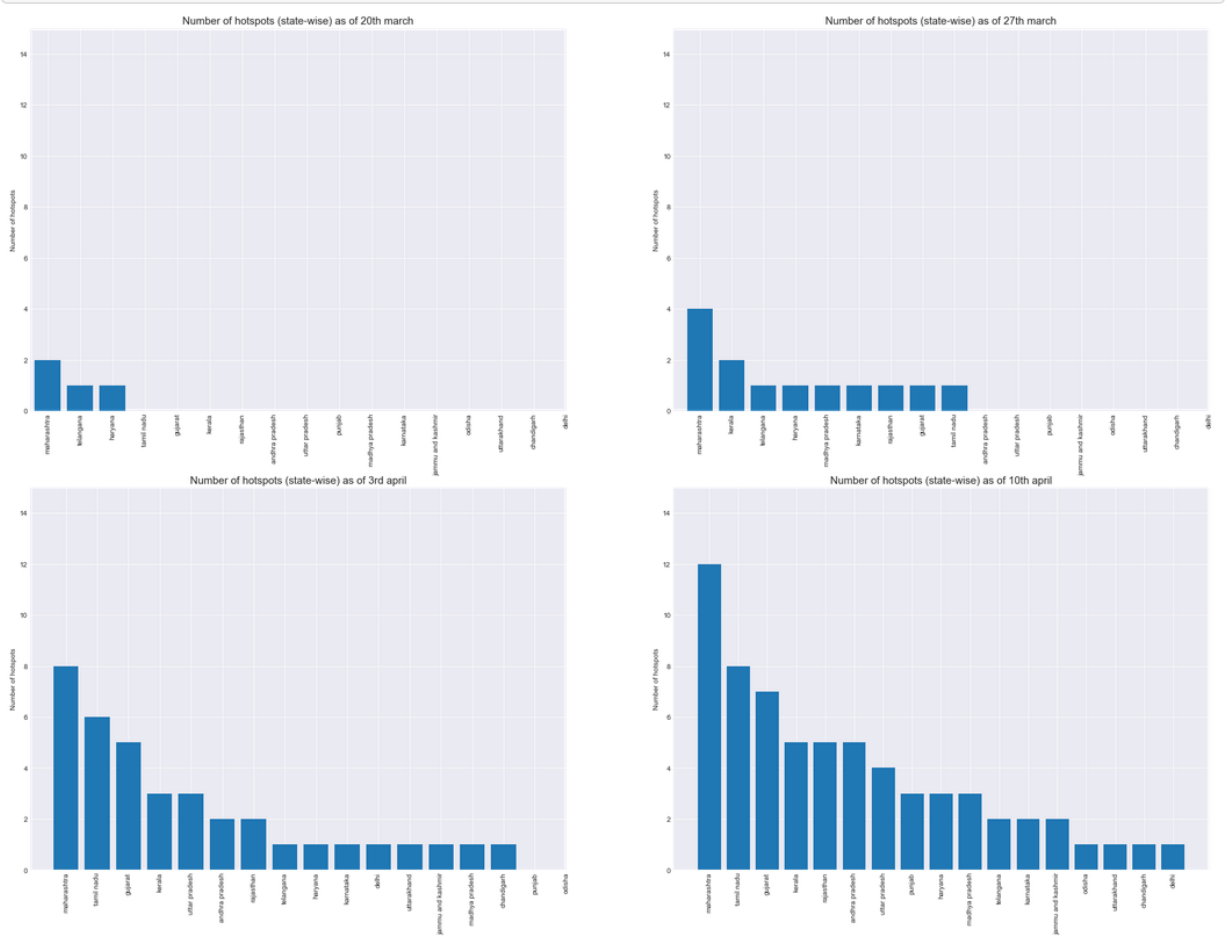
3.5

Duration	Maximum increase in hotspots	Maximum decrease in hotspots
20/03/2020-27/03/2020	Maharashtra (2)	Nil
27/03/2020-03/04/2020	Tamil Nadu (5)	Nil
03/04/2020-10/04/2020	Maharashtra (4)	Nil

Maharashtra has reported the highest surge in hotspots consecutively for the three weeks while no states showed a decline.

Here's how we proceeded in calculating the number of hotspots:
For example, for the week 28th March-3rd April: We are taking into the active cases count those who are diagnosed on or before 3rd April and in "hospitalized" condition and also those who are diagnosed on or before 3rd April, but recovered/died on any date AFTER 3rd April(this is examined by the 'status change' column). Via the second condition, we are checking for people who may have currently recovered/deceased but were hospitalized on or before 3rd April.

Given below is a graph showing the number of hotspots for the top 12 states:



3.6

We will categorise the cases based on the following observations in the notes section of the IndividualDetails.csv file:

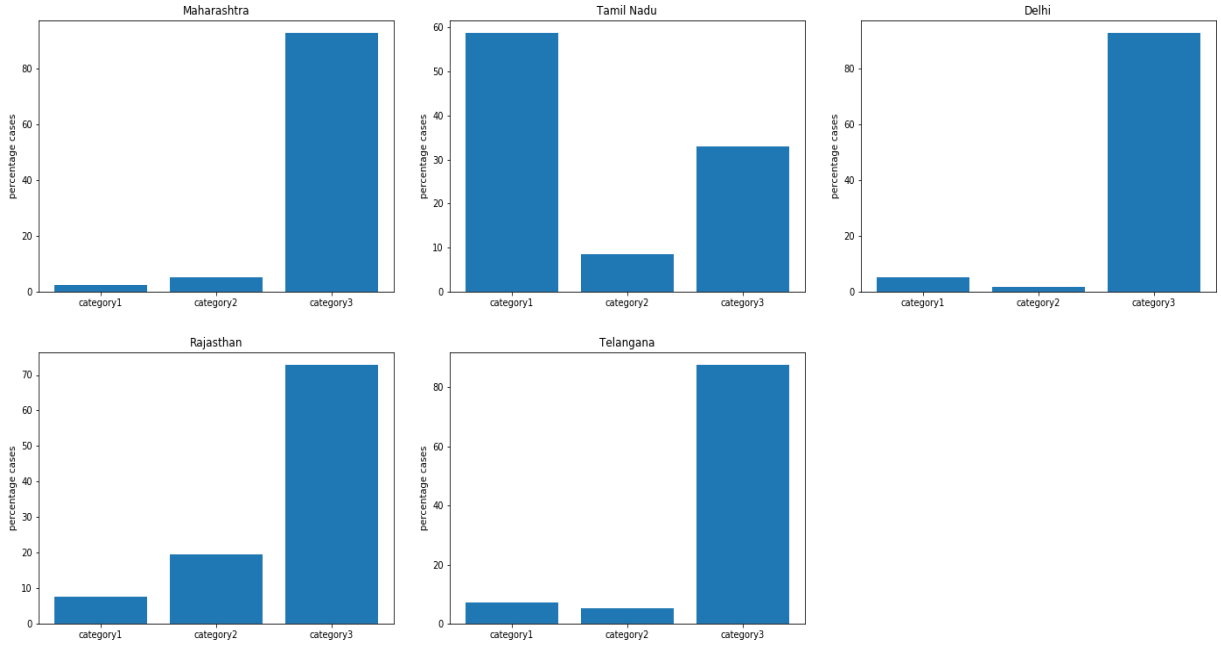
- Category 1: Cases with international travel generally have “travelled from” in the corresponding ‘notes’ column. We exploited this to form a **set of foreign places** in the data. We manually examined whether every place in the set is foreign.
- Category 2: Cases with personal contact with a primary case can be identified based on keywords on relationships (wife/daughter etc.) or case number (p52, 44). To form a list of relationships we have searched for words which precede ‘of’ and included those which made sense.
- Category 3: Cases which fall in neither of the above two categories.

To summarise, a case is primary if the notes section contains any place from the set of places. A case is secondary if it contains any of the relationships or mentions some other infected case (in the form of p followed by a number). We classify it as tertiary otherwise.

Percentage of different categories in top 5 with maximum number of cases

	detected_state	category1	category2	category3	cases
0	Maharashtra	2.541296	5.019060	92.439644	1574.0
1	Tamil Nadu	58.616905	8.452250	32.930845	911.0
2	Delhi	5.315615	1.882614	92.801772	903.0
3	Rajasthan	7.664884	19.607843	72.727273	561.0
4	Telangana	7.186858	5.338809	87.474333	487.0

Plots on different categories of cases

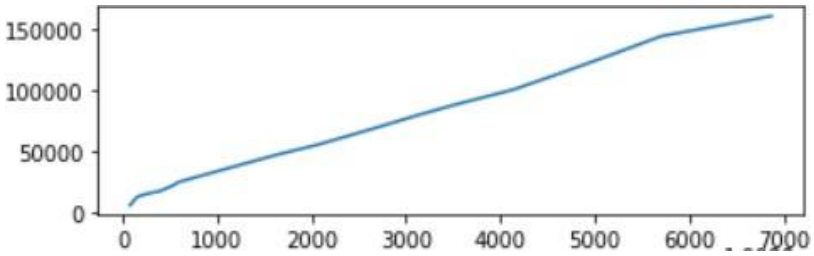


From the table, it is evident that Maharashtra and Delhi have reported more than 90% tertiary cases. This is quite worrying and contact tracing should be done extensively to prevent community transmission.

Also, we feel that there's a strong possibility of misclassification from our side because our criteria are not 100% effective in terms of coverage. Nonetheless, we learnt an important lesson that **it is important to maintain structured data** in order to derive the maximum insights. For example we could have an additional column explicitly classifying the case as primary, secondary or tertiary.

3.7

The data is analysed by plotting total samples tested v/s total positive cases detected till April 10th.

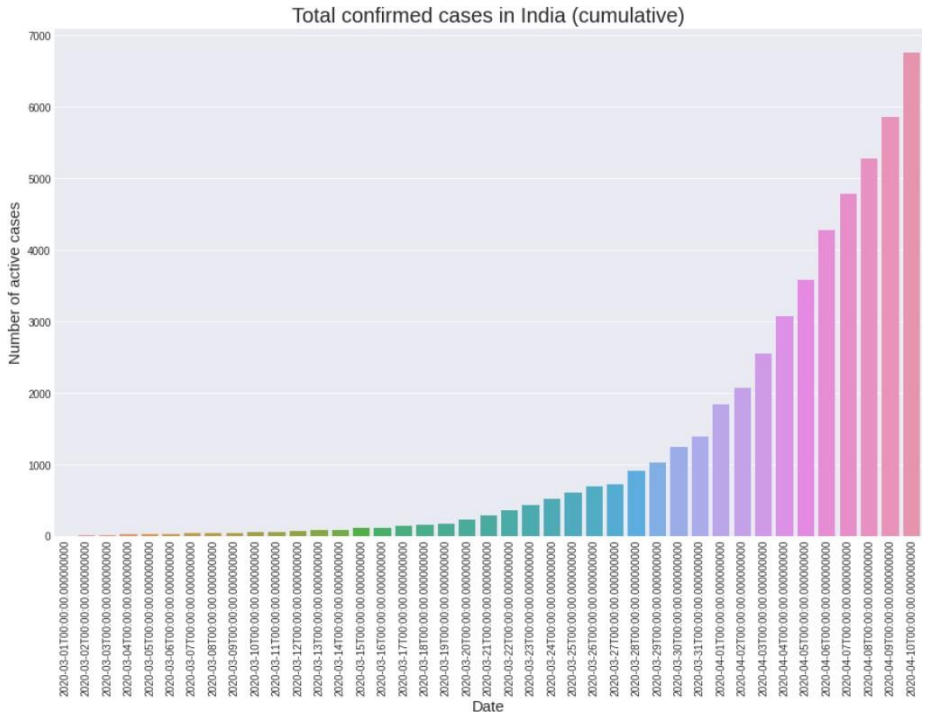


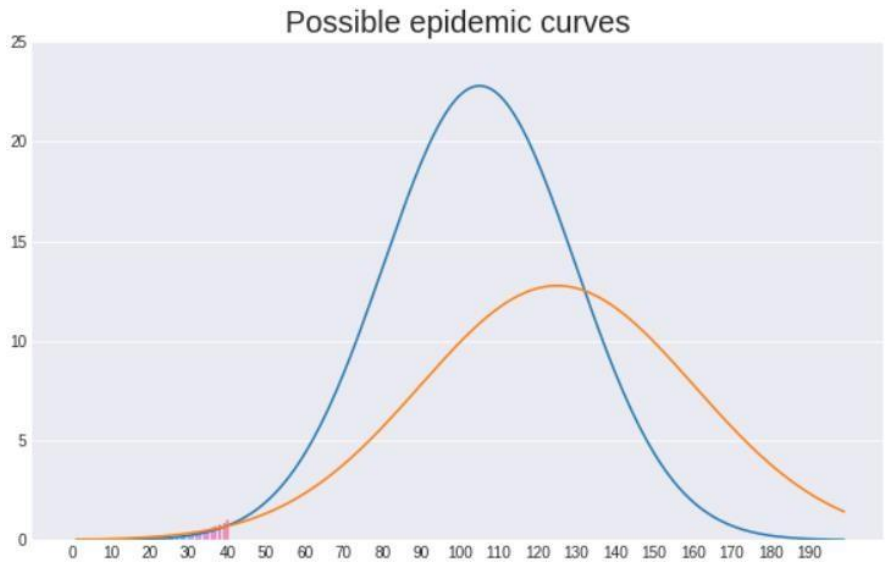
From the graph, it is perceivable that we have a **strong** Linear Correlation between the features. So we can build a Linear Regression model to predict the cumulative number of tests given the cumulative number of cases. The changes in number of cases can be assumed to be as given in the question.

Since we don't have the data of total number of labs, we will assume that all the labs were used in full capacity on the **day the maximum number of labs were used**. This could be found out by subtracting the cumulative number of samples tested, dividing by 100 and finding the max value.

Dates	11/04	12/04	13/04	14/04	15/04	16/04	17/04	18/04	19/04	20/04
Extra labs required	0	0	0	1	22	44	69	96	126	159

3.8



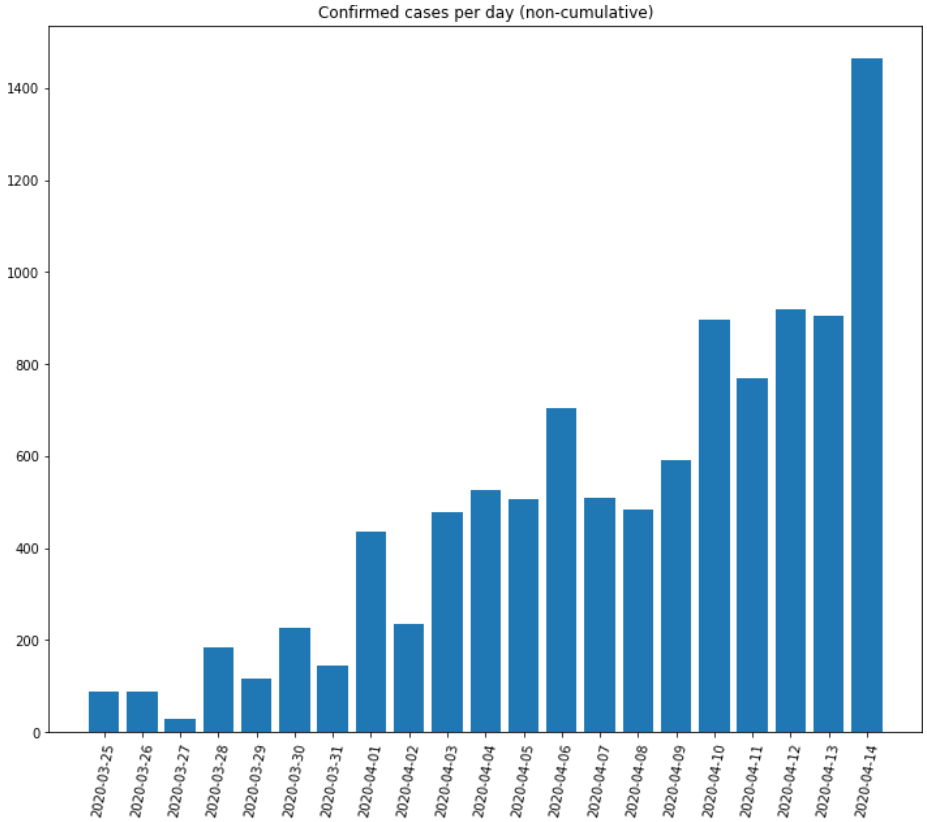


We haven't come to the phase of flattening the curve yet. From the fit Gaussian curves we can see that we are still in the **increasing** phase, we have not reached the peak yet.

We have fit two Gaussian curves to the normalized data to analyze this situation (x-axis is number of days with 01/03/2020 as 0). A steeper curve is the curve which fits the data better and has a taller peak, and a broader/flatter curve which has a shorter peak fits worse when we compare their root mean square errors.
From this we can see that if we follow the lockdown practices better, we can effectively flatten the curve and follow the green curve (but the peak will come much later). However on the other hand if we continue on the same rate and follow the steeper curve, we would end up with the peak sooner but will have **a lot more number of covid-19** cases.

3.9

A plot of confirmed cases per day has been made to see how effective the lockdown has been. The plot is not cumulative as cumulative confirmed cases per day will not give an useful idea on the effectiveness of lockdown.



We have verified the validity of our above graph with this link.

Note: Only the **dataset given** is used. The above link is for the purpose verification alone.
From the graph above we can conclude that the 21-day lockdown from 25th march to 14th April is only **partially successful** as by the end of lockdown i.e.14th April, the new infected cases have soared up (probably because people are not following it strictly).

We do see some relative dips in new infected cases on dates like 8th April and 11th April indicating that lockdown is indeed working.

Suggestion/Inference:

The lockdown **needs** to be (infact will be) extended beyond 14th April and more strict measures should be taken during the extended lockdown.

CH5019 PROJECT JAN-MAY 2020 **GROUP –3**

Narendhiran	CH18B015
Vishal S	CH18B020
Vignesh Kumar	CH18B118
Surya Narayan	MM18B036
Vishnu Kiran	CS18B067