QUESTION 1

Data Pre-Processing

For the first question, we have used MATLAB as a programming tool as it offers several inbuilt functions which would be useful to us and eliminates the need of downloading external libraries.

We transferred all the images to one folder and renamed them as “subject number (image number)” for easy access to all the images. After that we vectorized all images from 64 x 64 to 4096 x 1, and concatenated all 150 together into 1 dataset, in the order of the subjects and images given to us. This dataset was then split into 15 parts each containing all the images of a subject. We have defined a function which will take in a dataset containing all concatenated images of a subject (let’s call this matrix A.) and return a single representative image as an output.

Upon researching and calculating, we came up with the following method to solve the problem in the question.

Our Method

In this method, we have followed the following steps:

Following is a brief explanation /incentive for the steps followed in the algorithm.

- **Step 1:** We have first centralised the data matrix by subtracting the mean of all images (all columns) from each image (column) of the matrix. We call this centralised matrix $\Phi_{4096x10}$. This matrix can be considered as a feature matrix for a subject.

- **Step 2:** Then, instead of calculating the covariance matrix of dimensions 4096 x 4096 directly (which would be computationally very expensive), we have carried out the following calculation:

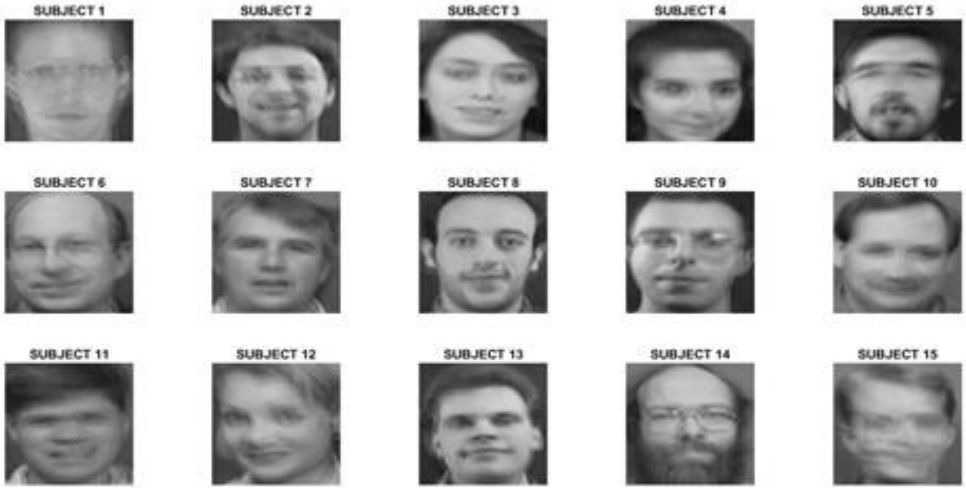
$$C=\Phi' * \Phi/10$$

This gives us a 10 x 10 matrix which is much easier and faster to compute.

Also, all the important information can be recovered later on using this.

- **Step 3:** We performed SVD on the above matrix to get standard U_{10x10}, S_{10x10} & V_{10x10} matrices. We further used the left singular vector to construct the 4096 x 1 vector that will be reshaped into the single image for each subject.
- **Step 4:** To do this, we first multiplied the data matrix $\Phi_{4096x10}$ with the first column of U_{10x10}. We only used the first column as it represents the most important features of the subject because it corresponds to the largest singular value. Then, we normalize the vector and multiply it with the corresponding eigenvalue (square root of the first singular value). At last, we added the mean of A back to the final vector.
- **Step 5:** Finally, we unrolled the final 4096x1 vector into a 64x64 matrix to get the representative image.

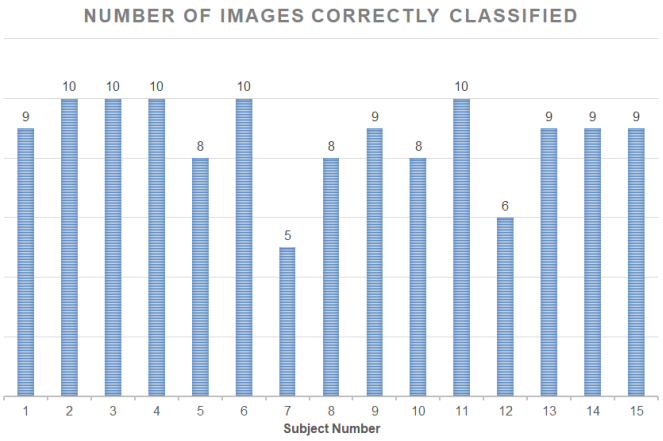
This method gave us an **accuracy of 130** (out of 150) while considering the minimum norm as judging criteria. We have achieved the following graphs and images for this method:



To validate this method, we carried out a simple experiment. We created a representative image using only 5 of the 10 images of a particular subject and tested the accuracy of the image on all 10. This can be seen as a test-train split. After doing this, we achieved an accuracy in the vicinity of 125, which means that the algorithm works on new data as well.

On the side, we have provided a histogram in which we have shown the number of images correctly identified for a subject based on the least norm. We can see that 13 out of 15 subjects have an accuracy of more than 80%, and it is the 7th and 12th subjects that are responsible for losing 9 points in accuracy.

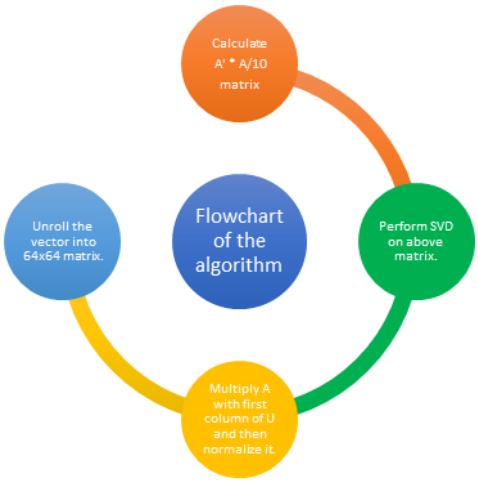
So we can conclude that this algorithm uses features from each image and outputs a representative image using SVD while giving satisfactory results even on new data. Thus, this algorithm would serve its purpose.



But there was **one more algorithm** which gave mind-blowing results and hence we believe, is worth mentioning.

This algorithm works on *a basic principle of dimensionality reduction*, where one representative image is obtained using 10 different images of a subject. We wanted to try another method so that we can compare the solutions of both the algorithms, both accuracy wise and image wise. The steps for this algorithm are given below.

- In this method, we **did not** subtract the mean of 10 images from the data matrix $A_{4096x10}$.
- We performed SVD on $A' * A/10$ to get standard U_{10x10}, S_{10x10} & V_{10x10} matrices. We further used the first left singular vector to construct the 4096 x 1 vector that will be reshaped into the representative image for each subject.
- To do this, we first multiplied the data matrix $A_{4096x10}$ with the first column of U_{10x10}. We normalised the achieved vector and multiplied it with corresponding eigenvalue (square root of the first singular value).



- Finally, we unrolled the final 4096 x 1 vector into a 64 x 64 matrix to get the representative image.

Using this method, we have achieved an outstanding **accuracy of 149** (out of 150).

This method also delivers an accuracy in the range of 145 for the above-mentioned test-train split.

We have achieved the following images for this method:

Comparison

From the two methods given above, we can see that it is the first method in which we have carried out a feature extraction of the most important ones from the given data set, thus providing the answer to the problem statement. This is achieved by subtracting the mean from the data as it helps centralize the data and reduce the irregularities in the data. As compared to this, the other algorithm mentioned above operates directly on the data matrix as this was based on a dimension reduction theory.

After taking a look at the results both of them gave, we can see that although the accuracy of the second algorithm is very high, the images produced by it (for some subjects) are barely even recognizable as faces. Just as an experiment, we compared the images from this method to the mean of the faces of each subject, and we found out that the pictures are almost the same. This could be the reason that some faces (especially the ones whose photos are taken from different angles) are blurry. But for the first algorithm, although the accuracy is comparatively low, the faces look much better to the human eye as compared to the other ones. We attribute this to the fact that we took all the important features of the image and used them to create one. We believe that these faces will perform better if an image of a subject in a different environment is given, as it works on the features of the face rather than a simple accumulation of all the images into one.

List of References:

- http://www.vision.jhu.edu/teaching/vision08/Handouts/case_study_pca1.pdf
- https://sites.cs.ucsb.edu/~mturk/Papers/mturk-CVPR91.pdf
- https://medium.com/@rajnair.ds/an-intuitive-study-of-pca-and-svd-with-the-help-of-co-variance-and-eigen-vectors-5e7eece62634
- https://towardsdatascience.com/eigenfaces-recovering-humans-from-ghosts-17606c328184

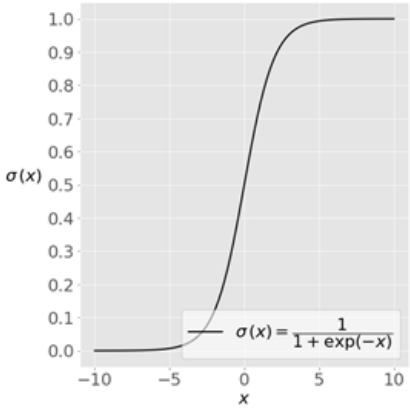
QUESTION 2

Introduction

Logistic regression is a widely used statistical classification model. It uses a logistic function to model a binary dependent variable. In this project, logistic regression is used to predict whether a reactor will operate under a given set of conditions.

Methodology

We implement L2 regularized logistic regression model using gradient descent as the optimization method. Logistic regression is a linear classifier, so we use a linear function $f(\mathbf{x}) = b_0 + b_1x_1 + \cdots + b_rx_r$, also called the **logit**. The variables b_0, b_1, …, b_r are the **estimators** of the regression coefficients.



The logistic regression function $p(\mathbf{x})$ (shown in Fig 1) is the sigmoid function of $f(\mathbf{x})$: $p(\mathbf{x})$ = 1/ (1+exp(−$f(\mathbf{x})$). Therefore, it's often close to either 0 or 1. The function $p(\mathbf{x})$ is often interpreted as the predicted probability that the output for a given $\mathbf{x}$ is equal to 1. Therefore, 1 − $p(x)$ is the probability that the output is 0.

Dataset

The given dataset consists of parameters that may control whether a reactor will operate or not. The dataset has 1000 rows of data. The final column of the dataset gives the output as 'Pass' or 'Fail' and is named as the 'Test' column.

	Temperature	Pressure	Feed Flow rate	Coolant Flow rate	Inlet reactant concentration	Test
0	406.86	17.66	121.83	2109.20	0.1033	Pass
1	693.39	24.66	133.18	3138.96	0.3785	Pass
2	523.10	23.23	146.55	1058.24	0.4799	Fail
3	612.86	40.97	94.44	1325.12	0.3147	Fail
4	500.28	37.44	185.48	2474.51	0.2284	Pass

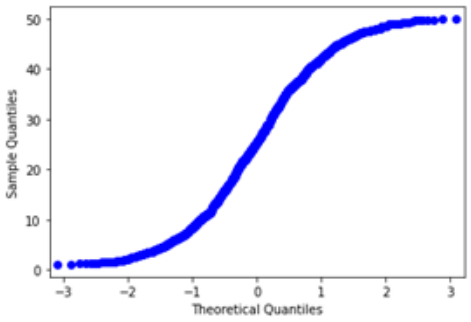
Data Analysis

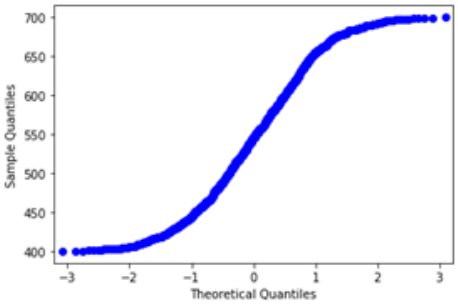
Our dataset is multivariate, i.e. has more than one independent variable. Therefore, the statistics of the given data is visualized using multiple analysis tools such as:

Quantile-Quantile plot, Correlation matrix, Calculation of mean and standard deviation for each variable, Scatter plot, Seaborn's Heatmap and Pairplot.

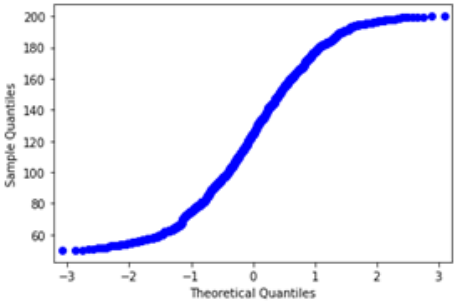
Quantile-Quantile plot

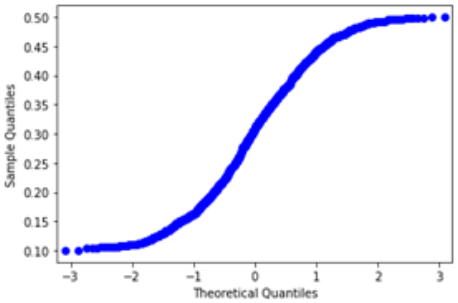
We used Q-Q plots since they reveal whether the given data belongs to a theoretical distribution.

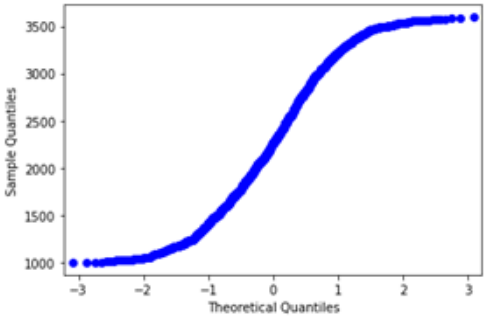




. Temperature Pressure



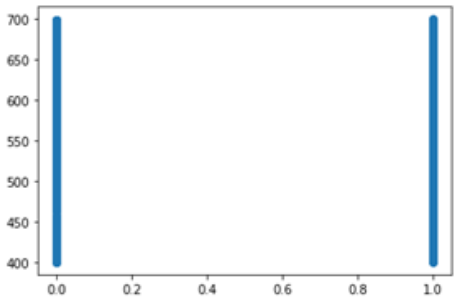




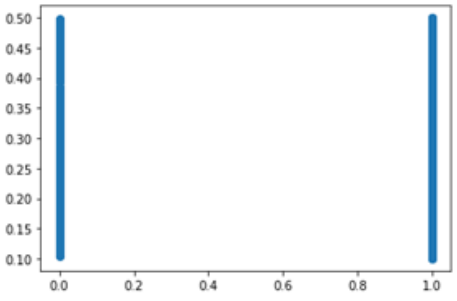
Feed Flow Rate Coolant Flow Rate Inlet Reactant Concentration

We note that the Q-Q plots are s shaped curves. On a Q-Q plot under-dispersed data appears S shaped. Under-dispersed data has a reduced number of outliers (i.e. **the distribution has thinner tails than a normal distribution**). Hence, we conclude from the Q-Q plots that our variables have **under-dispersed data**.

Scatter Plots



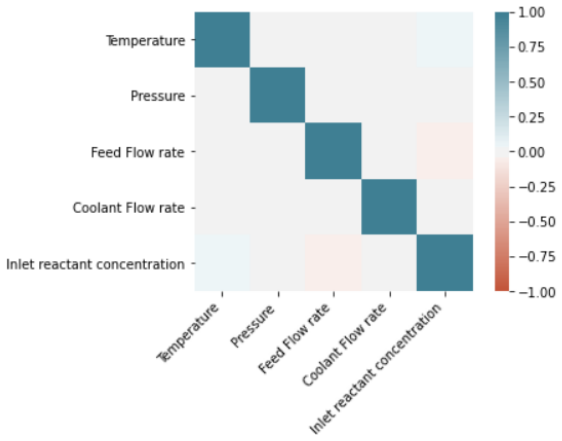
Inlet reactant concentration Test



Temperature – Test

The statistical parameters such as **mean and standard deviation of pass data and fail data were found to be similar in values**, and hence more such scatter plots were not drawn.

Correlation Matrix



Here, we notice a slight negative correlation between Inlet reactant concentration and Feed flow rate, while there is a slight positive correlation between Inlet reactant concentration and Temperature.

Data statistics

Mean

```
Standard deviation of data of each variable
 Temperature                      86.858780
Pressure                          14.252407
Feed Flow rate                    43.508159
Coolant Flow rate                763.680625
Inlet reactant concentration       0.116062
Test                               0.492969
dtype: float64
```

Standard Deviation

```
Mean of data of each variable
 Temperature                     546.766430
Pressure                          25.493270
Feed Flow rate                   125.029060
Coolant Flow rate               2295.797770
Inlet reactant concentration       0.302692
Test                               0.585000
dtype: float64
```

Design and Analysis of Algorithm

The algorithm has been realized in Python because of its readability and ease of use. The libraries imported are: **NumPy, Pandas, MatplotLib, Seaborn, StatsModel, Tqdm** (for progress check).

Data Pre-processing

Data randomization. The training examples given may not be in random order, which may produce misleading results. Therefore, we need to randomize the dataset first before dividing it into training subset and testing subset.

Data quantification. Data given in the last column ('Test') is of binary classification type. Hence, we convert 'Pass' as 1 and 'Fail' as 0.

Data standardization. We needed to standardize the data before putting it in our model. For this we used the formula (x-mean)/std. Now the train data had mean 0 and standard deviation=1. The test data was transformed using the same formula but values of mean and standard deviation were taken from train data such that statistics of train data was reflected in test data.

This process is necessary since our dataset had values which were quite large, and if we pass those values through sigmoid it would just output -1 and we wouldn't make any progress.

Train/Test split. Dataset is split into a training set and testing set. 70% of the dataset is used as training data, and the rest 30% is used as test data.

Model Fitting

Logistic regression determines the best predicted weights $b_0, b_1, \ldots, b_r$ such that the function $p(\mathbf{x})$ is as close as possible to all actual responses y_i, $i = 1, \ldots, n$, where n is the number of observations. The process of calculating the best weights using available observations is called **model training** or **fitting**.

To get the best weights, our algorithm maximizes the **log-likelihood function (LLF)** for all observations $i = 1, \ldots, n$. This method is called the **maximum likelihood estimation** and is represented by the equation LLF = $\Sigma_i(y_i \log(p(\mathbf{x}_i)) + (1 - y_i) \log(1 - p(\mathbf{x}_i)))$.

Iteration was performed multiple times, but after 40 epochs, the model started to overfit the data and hence accuracy dropped after 40 epochs.

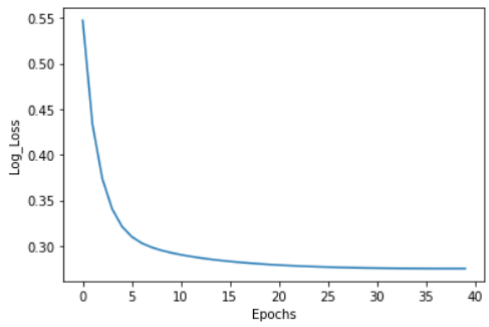
Model Quality Assessment

Log Loss. It is a function that quantifies the accuracy of a classifier by penalising false classifications. Minimising the log loss is basically equivalent to maximising the accuracy of the classifier.

The reason we chose log loss/cross entropy over Mean square error (MSE) is as it is a better measure for classification. This is because the **decision boundary in a classification task is large** (in comparison with linear regression). MSE doesn't punish misclassifications enough but is the right loss for linear regression, where **The distance between two values that can be predicted is small.**

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

Also, from a probabilistic point of view, the log loss arises as the natural cost function to use when we have a **Sigmoid or SoftMax nonlinearity in the output layer of our network**, and we have to **maximize the likelihood of classifying the input data correctly**. The formula for Log Loss is given below:



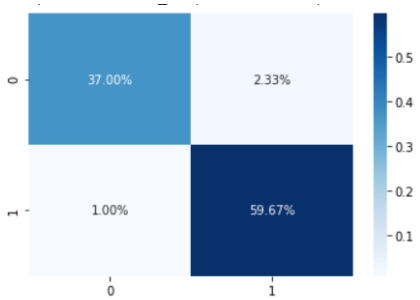

```
[528] print(accuracy_train, accuracy_test)

      94.42857142857143 96.66666666666667
```

From Fig, we can observe that the log loss is getting minimized and hence the model is working properly.

Confusion matrix. This matrix classifies the predictions into four categories. It helps us calculate the parameters such as precision and recall. Precision and recall help assess the quality of a model. Confusion matrix also helps in visualizing what percentage of the data has been classified correctly.

Both type 1 and type 2 errors are below 5%. If we decrease either of the two errors by cutting out some data, we will needlessly increase the other error too.

Reported Confusion Matrix



Reported Model Assessment Parameters

```
confusion_matrix
[[111.   7.]
 [  3. 179.]]
precision 0.9623655913978495
recall 0.9835164835164835
f1 0.9728260869565217
```

It is the harmonic mean of precision and recall, both of which are obtained from the confusion matrix. It is high only when both the values are high and therefore can be used in lieu of them both. It's value is preferred to be close to 1.

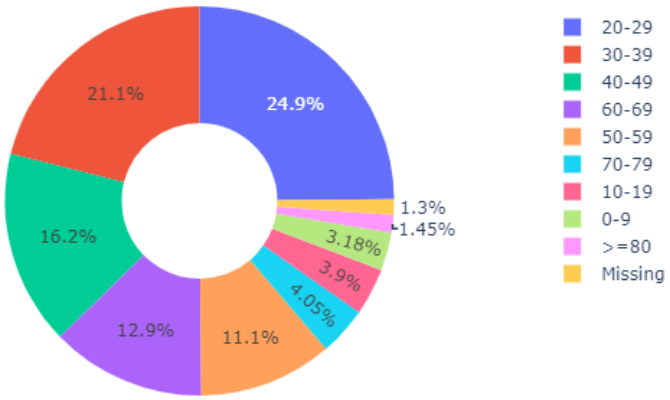
For our model, reported **F1 score = 0.9728260869565217**

QUESTION 3

The first case of Covid-19 in India was found in Kerala on 30th January, 2020. Since then India has been fighting to restrain the growth of the virus. This report is an effort to analyse India's fight against Coronavirus using data.

Age wise distribution of cases

From the available data, a pie chart could be formed which shows most of the cases are from the age group 20-29. This fact is rather not surprising considering the facts that the epidemic is still in its stage 2, thus mainly affecting working class citizens and the median age in India is about 27 years old. Following this age group are the age groups 30-39 and 40-49 which further accentuates the susceptibility of the working class to the virus.
Agedetails.csv was used to make this chart. Since the missing data is only about 1%, data is relatively clean. This section solves question 1.

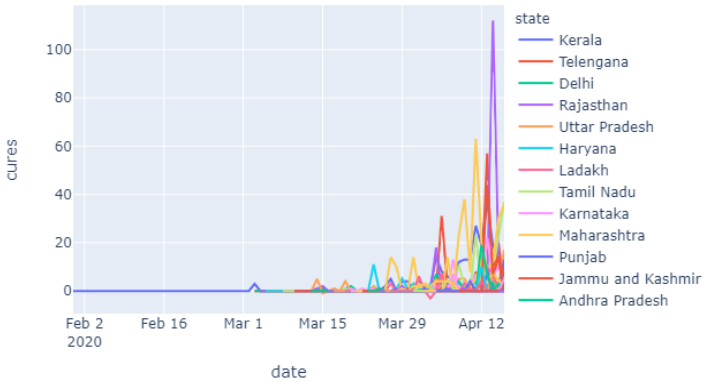


Daily trend of cases

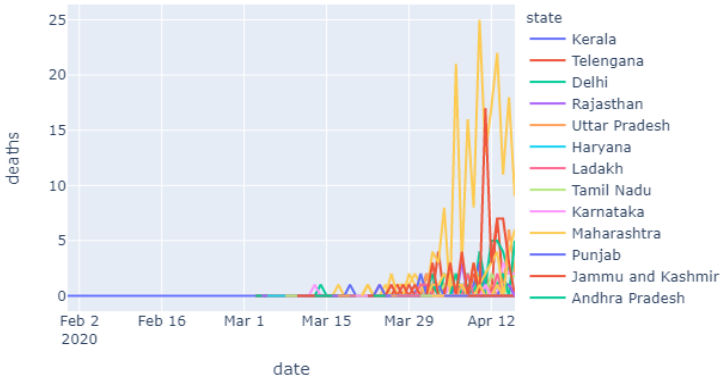
The following graph shows the number of cases observed per day in each state. A pandemic is an example of an exponential model. Thus, the number of cases should increase every day exponentially. But one could observe a few spikes in the graph.
These spikes indicate a special event that might have happened that day which catalysed the community spread of the virus.

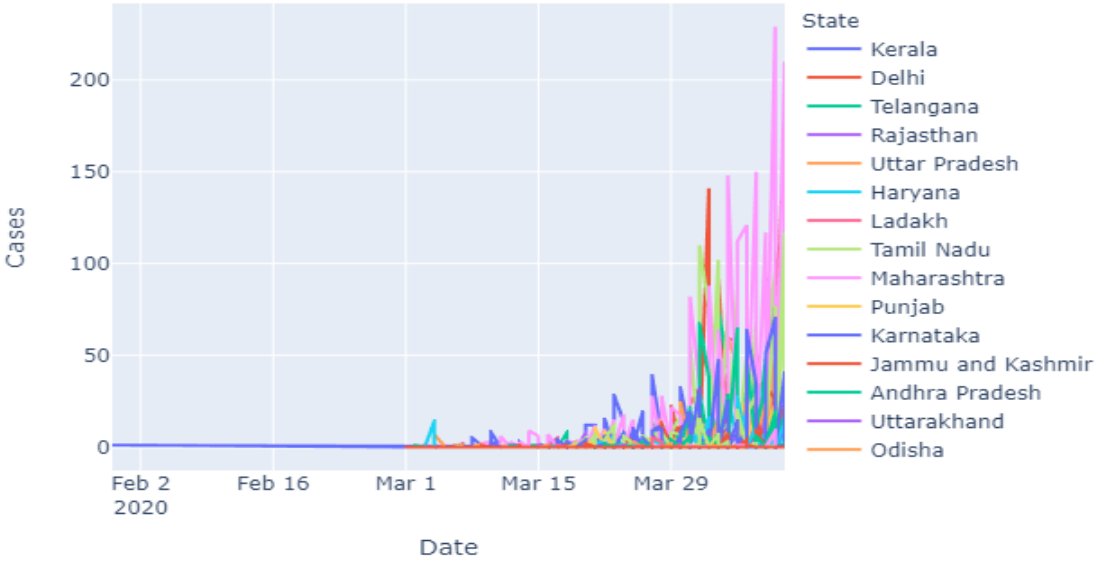
This data was generated using IndividualDetails.csv file grouping the data by state and sorting by date. This section solves question 2.

Cures per day per state

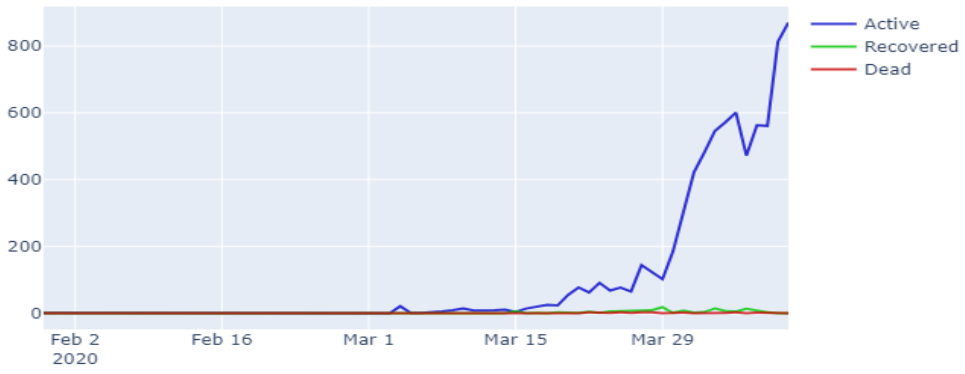


Deaths per day per state





Cumulative graphs of the cases recovered, deceased and observed are depicted in the following graph which shows the trend in cases on a daily basis.



Hotspots in the country

Although the above graph depicts the number of cases per state. It does not depict how well the virus is controlled in that particular state. This is attributed to the R0 number or the basic reproduction number. For a pandemic, the R0 number tells us about how infectious the disease is. The value of R0 for Covid-19 is around 2.2. The more the people an infected person meets, the more is the probability of the virus spreading.

Thus, population density is one more factor to consider. A new term could be introduced as a measure of the outbreak which is - Intensity. Intensity of the outbreak is defined by the number of cases divided by the population density of the state. The following chart shows the number of cases and intensity for each state.

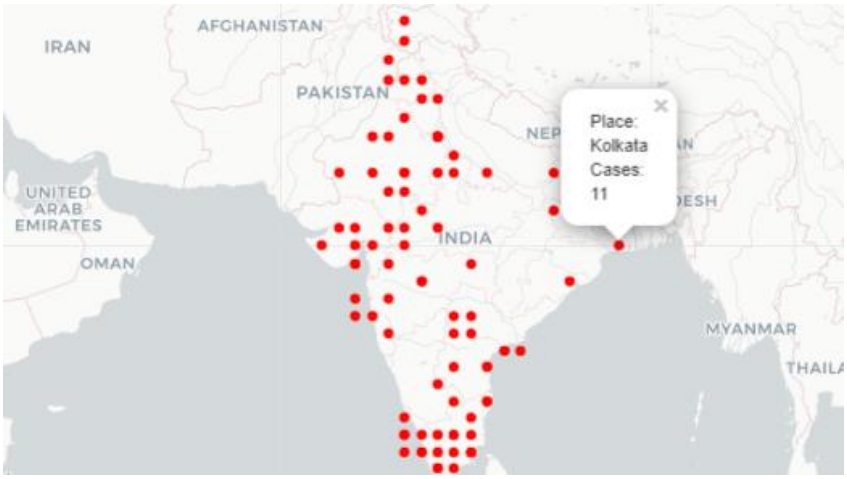
The following table was generated using the background gradient attribute of pandas library. This section solves question 3.

	Cases	inensity
Maharashtra	1574	4.31233
Tamil Nadu	911	1.64144
Delhi	903	0.0799327
Rajasthan	561	2.79104
Telangana	487	1.5609
Madhya Pradesh	451	1.91102
Uttar Pradesh	433	0.522947
Andhra Pradesh	381	1.25743
Gujarat	378	1.22727
Kerala	363	0.422584
Karnataka	207	0.648903
Jammu and Kashmir	207	2.11224
Haryana	176	0.307155
Punjab	151	0.274545
West Bengal	116	0.112731
Bihar	60	0.0544465
Odisha	50	0.185874
Uttarakhand	35	0.185185
Assam	29	0.0730479
Himachal Pradesh	28	0.227642
Chandigarh	19	0.00205361
Chhattisgarh	18	0.0952381
Ladakh	15	5.35714
Jharkhand	14	0.0338164
Andaman and Nicobar Islands	11	0.23913
Goa	7	0.0177665
Puducherry	7	0.00269438
Manipur	2	0.0163934
Tripura	2	0.00571429
Mizoram	1	0.0192308
Arunachal Pradesh	1	0.0588235
Dadra and Nagar Haveli	1	0.00103093

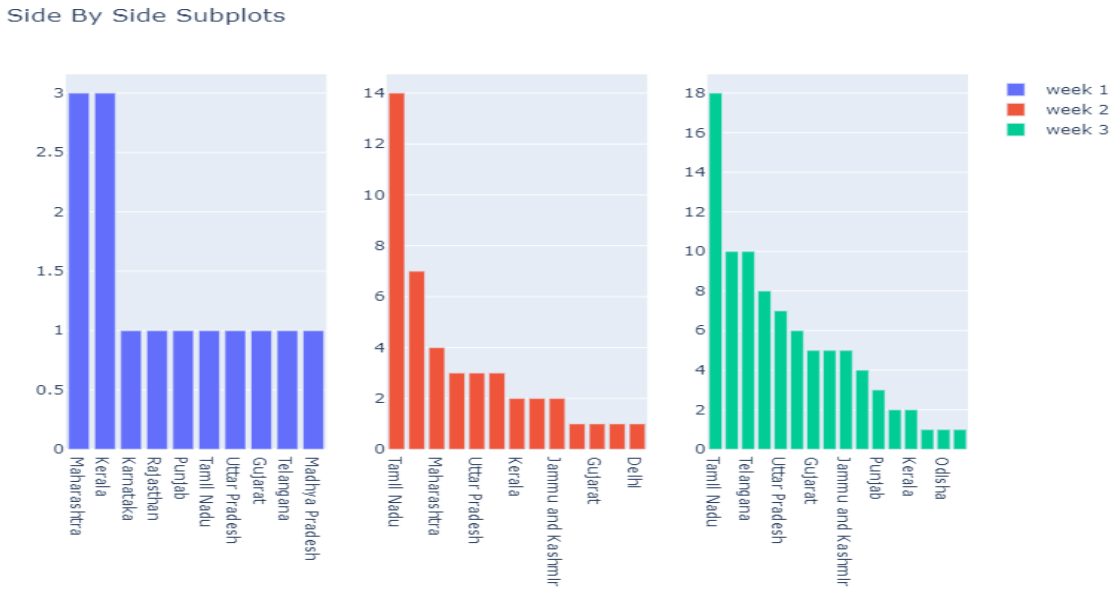
Since no vaccine has been found for the disease, the best way to prevent getting infected is to stay isolated. Thus, self-quarantine is the best way to fight the pandemic. But locking down the country affects the economy inordinately. Thus, we need to list down which places should be on a complete lockdown. We introduce a new term here - An epidemic hotspot. A hotspot is defined as a place with more than 10 cases.

If we parse the given data by cities, we get relatively less number of hotspots. Thus we decided to parse the data by districts. The following map shows the hotspots in the country.

IndivdualDetails.csv file was used to find the hotspots in the country. We got the coordinates for the hotspots from This Kaggle dataset. Folium library was used to plot the map. This section solves question 4.



The following graph shows the trend in the number of hotspots for a few states by week.



IndividualDetails.csv file was split into 3 weeks and used to plot this graph. This section solves question 5.

Investigation on the types of the cases

For the investigation on how the virus spread in the country, segregating the cases could be helpful. The cases could be classified into three categories, primary - ones with international travel history, secondary - ones who were in contact with primary cases, tertiary - every other case.

Now the data we had access to provided one column as notes in individualdetails.csv file. This column did not directly classify the patient as primary, secondary or tertiary. Going through this data manually would be a long and boring task. Thus, we made certain assumptions about the data. Lot of data was missing - i.e. the details were yet not received.

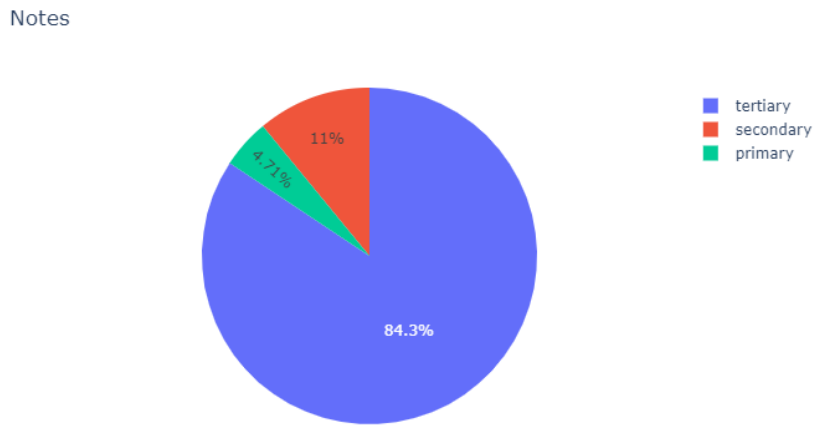
Going through the data, we noticed most of the patients with international travel history contained the word Travelled from. Analysing notes containing 'Travelled from', we noticed all such cases (192) except 4 were international cases. Thus, it is safe to say that if a note contains 'Travelled from', it's an international case.

As for secondary cases, we noticed a fair share of them contained the pattern 'Contact with *patient ID*'. Thus, we searched for the pattern 'Contact'. Also, a large share of secondary cases is from families. Thus, we searched for 'related', 'family', 'father', 'mother', etc.

The country was on a lockdown since 21st March. International passengers were being isolated. The virus has the incubation period of 15 days. Thus, we safely assumed that number of tertiary cases will increase, secondary cases will decrease and primary cases will be zero after certain time.

As for details awaited, we made some assumptions. We downloaded the dataset (cases till 10th April) on 17th April and 21st April. And ran our algorithm in both. The new dataset contained less entries with details awaited as expected. The entries which changed counted to 11. Out of which zero were primary, two were secondary and rest tertiary. The two secondary entries were diagnosed before 1st April. Thus, we assumed no secondary cases would be found after 31st March.

Therefore, we multiplied the number of entries with details awaited before 1st April by 2/11 and added them to the secondary cases class and rest to tertiary. Here's what we found from the final segregated data -



IndividualDetails.csv was used to solve question 6.

Additional labs needed

Assumptions: From 11th to 20th April, the cases increase at the rate of 10%. Each lab can perform 100 tests per day.

From the individualDetails.csv, we found the number of cases observed daily, which was 871 on the 10th of April. Total tests done on the same day were 1,61,330.

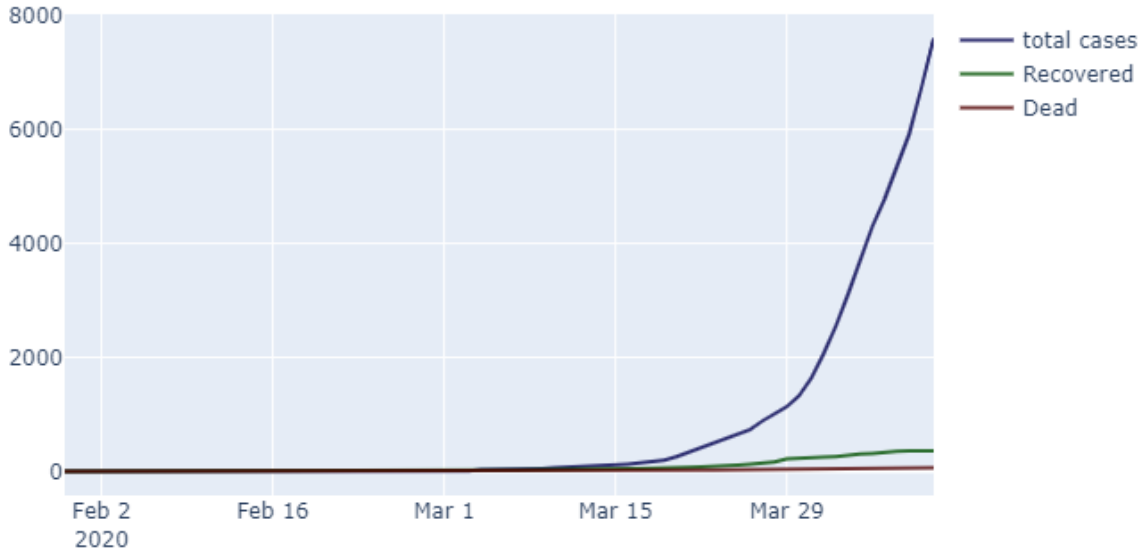
Now observing the labs data, one can notice that for one case, multiple tests are done. We calculated this factor between positive cases and observed daily cases.

Checking the new cases daily and the positive tests daily, we noticed that positive tests are 5-8 times the daily cases. For 10th April this number was 7.5, for 9th It was 7, for 8th It was 9, for 7th It was 7. Thus, we considered the factor of 7.5 for cases on 11th to 20th.

Thus, we first calculated the predicted number of observed cases for the period 11th to 20th April. Then we calculated the number of samples which should be positive, by multiplying the cases by 7.5. Then we calculated the total number of tests required by dividing positive cases by tests_to_positive ratio obtained from the labs data.
Thus, we found out that on 20th of April, we would need 2,73,258 total tests done. Subtracting 1,61,330 of 10th April leaves us with 1,11,928 additional tests. Dividing this number by 100, we conclude that we will need 1120 labs for successful detection of infected cases till 20th of April. *This section solves question 7.*

Fitting and forecasting time series data

The following graph depicts the observed cumulative cases as a time series data.



This section solves question 8.

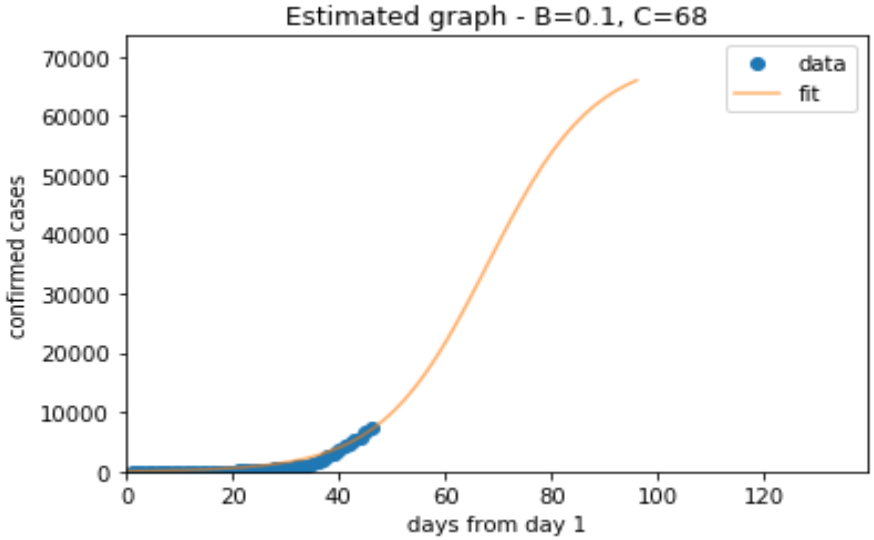
We went through the time series data of China and South Korea. Both of these countries have successfully managed to contain the growth of the virus. Analysing their data, we figured out that it resembled a sigmoid function. Thus, we tried to fit the curve for Indian data.

A sigmoid function has three parameters -
a - The limit for the cumulative number of cases.
b - Growth rate
c - x coordinate of the inflection point.

The popular notion of flattening the curve suggests that even if the cumulative number of cases is high, the cases spanning over a larger spectrum of time would be preferable. With respect to the sigmoid function, flattening the curve implies smaller b (growth rate) and larger c (inflection point)

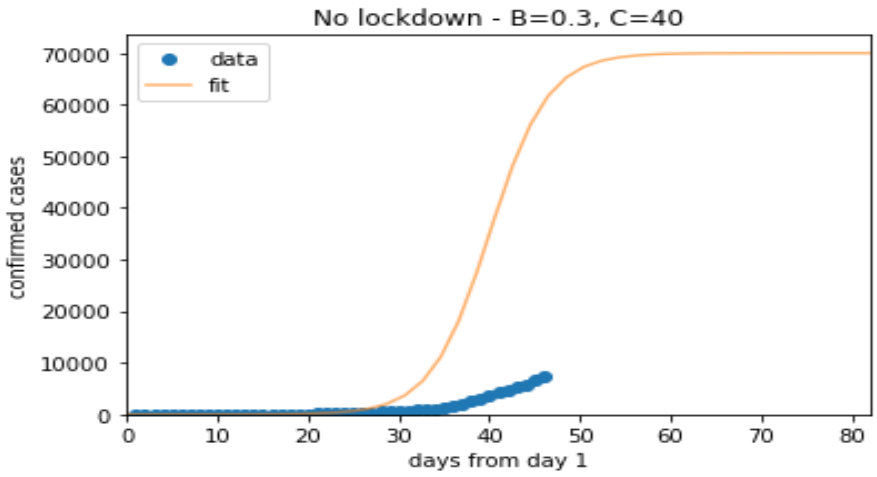
When we tried to fit the time series data for coronavirus cases in India into a sigmoid function, we found out the best parameters that fit the curve were,

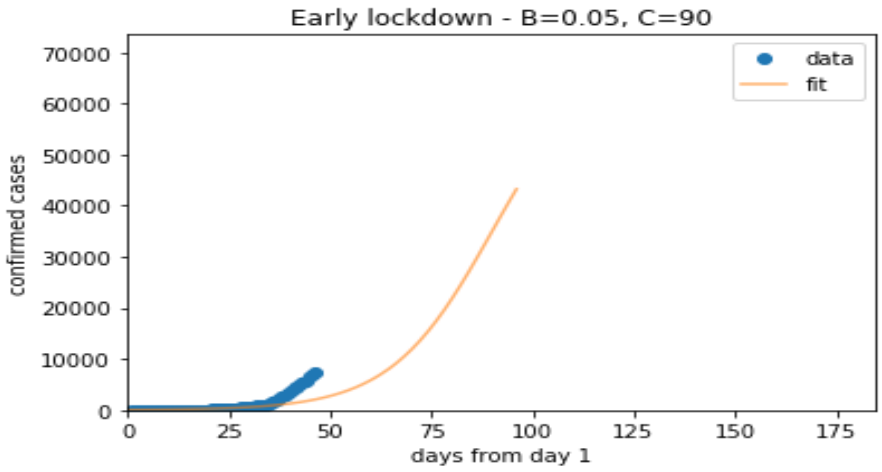
A = 70000 = Cumulative number of cases
B = 0.1 = Growth factor
C = 68 days after the first case. (The first case here is taken to be the case tested positive on the 2nd of March since the first few cases were isolated successfully and thus do not contribute to the model.)



To fit this curve, we used scipy and pylab libraries. We looked for graphs for other countries to decide the probable shape of the graph for India.

Now how the lockdown helps flattening the curve is by reducing the growth rate and pushing the inflection point further away. These are some possible scenarios had the lockdown not been implemented or had it been implemented earlier -





This section solves question 9.