CH5019:Mathematical Foundations of Data Science

May 15, 2020

Group no: 9

Roll No	Name
MM18B012	Anagha Belavadi S
CS18B002	Aniruddha Kawade
CH18B035	Gunavardhan

Roll No	Name
CH18B067	Shania Mitra
CS18B054	Dipesh Tandel

Question 1

Face Recognition :

Grayscale Images of different subjects are given and each subject has images obtained under different circumstances. The Facial Recognition method used is based on SVD-based projection of images which requires less memory compared to other facial recognition techniques.

Each subject is represented by a representative image and the facial recognition for a test image is based on the **minimum L1 norm** between the test image and representative image among all representative image. The representative image is obtained by extracting characteristic features from each image of a subject.

Number of subjects - 15

Number of images for each subject - 10

Each grayscale image (*given as .pgm file*) is stored as a 64×64 data matrix where each entry of the matrix is a value from 0-255 which represents the intensity of the respective pixel.

Algorithm -

Let the gray level images of size 64×64 of j^{th} subject out of 15 subjects be $F_1^{(j)}$, $F_2^{(j)}$, $F_3^{(j)}$, ..., $F_{10}^{(j)} \in \mathrm{R}^{64\times64}$ with $1 \leq j \leq 15$.

1. Convert all 64×64 images into images of size 1×4096.
2. For each image F_i^j $1 \leq i \leq 10$ and $1 \leq j \leq 15$ construct an image S_i^j such that $S_i^j = (F_i^j$ - $mean(F_i^j))/std(F_i^j)$.
   (a) $mean('matrix')$ returns a matrix of same dimension with all entries as mean of the values in the $'matrix'$.
   (b) $std('matrix')$ returns the standard deviation of $'matrix'$.
3. For each subject (*say* j^{th} *subject*) calculate reprsentative image R_j -
   (a) Construct P_j, 10×4096 matrix, where each row represents an image of the j^{th} subject (S_i^j $1 \leq i \leq 10$).
   (b) Apply SVD on P_j and obtain U_j.
   $P_j = U_j \Sigma_j V_j^T$
   (c) Evaluate R_j as $R_j = (P_j^T U_{j3})^T \in \mathrm{R}^{1\times4096}$ where $U_{j3} \in \mathrm{R}^{10\times1}$ is the sum of first 3 columns of U_j.
4. For every test image T -
   (a) Convert T into 1×4096 size matrix and compute $Y = (T$ - $mean(T))/std(T)$
   (b) Compute the norm of $|Y - R_j|$ $1 \leq j \leq 15$ for each representative image.
   (c) Find the minimum norm and return the subject corresponding to the representative image having minimum norm.

(d) If the subject returned is the actual subject then match is successful otherwise a match is unsuccessful.

Representative image of all subjects -

Performance -

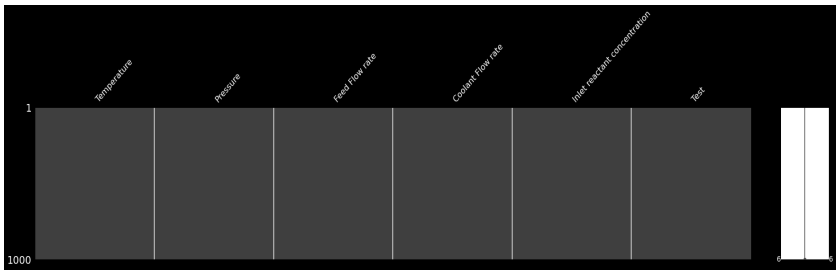
Number of successful matches(s) = 117

Number of unsuccessful matches(u) = 33

Accuracy = $(s/(s+u))\ 100$ = 78.0 %

Question 2

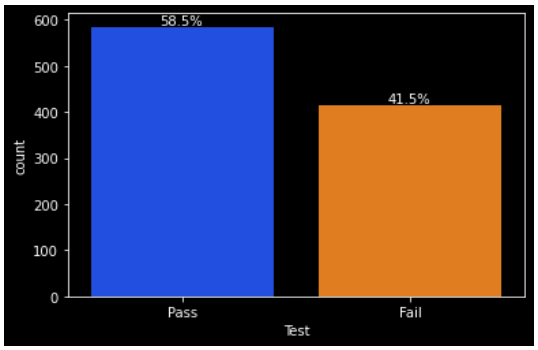
- As can be seen in the plots below,there is **no missing data** to be accounted for.



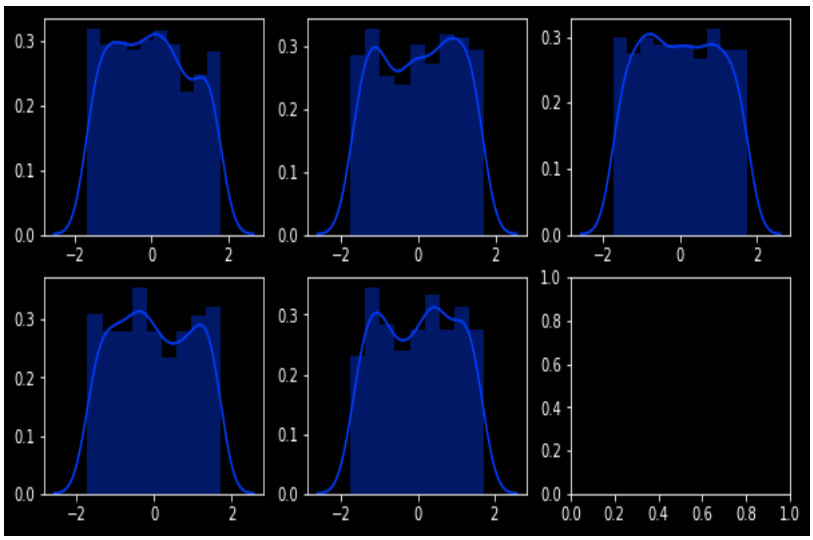

```
Temperature                    0
Pressure                       0
Feed Flow rate                 0
Coolant Flow rate              0
Inlet reactant concentration   0
Test                           0
dtype: int64
```

- This table gives the minimum, maximum, mean, standard deviation and the quartiles of the data. The data has **1000 samples** and **5 features**. All the features have **negative kurtosis** which indicates that the distribution has lighter tails and a flatter peak than the normal distribution (**under-dispersed distribution**) All features have skewness between between $-\frac{1}{2}$ and $+\frac{1}{2}$. hence the distribution is **approximately symmetric**. To remove outliers, we clipped the values between **1** and **99** percentile.

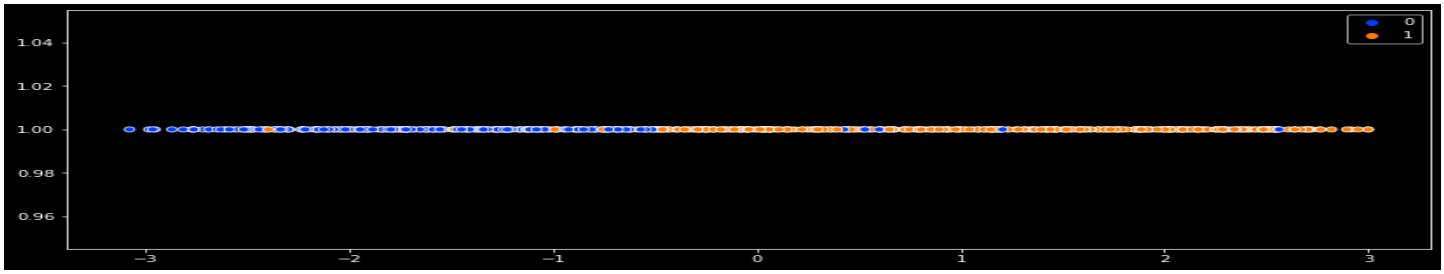
	Temperature	Pressure	Feed Flow rate	Coolant Flow rate	Inlet reactant concentration
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000
mean	546.766430	25.493270	125.029060	2295.797770	0.302692
std	86.858780	14.252407	43.508159	763.680625	0.116062
min	400.310000	1.060000	50.030000	1002.530000	0.100300
25%	469.735000	12.725000	88.587500	1635.682500	0.199075
50%	545.800000	25.375000	124.590000	2268.710000	0.308850
75%	618.877500	37.820000	162.562500	2983.692500	0.401625
max	699.870000	49.890000	199.960000	3595.620000	0.499600
variance	7544.447741	203.131114	1892.959914	583208.097420	0.013470
skewness	0.069251	-0.021071	0.018500	0.037458	-0.025999
kurtosis	-1.178749	-1.256784	-1.216024	-1.230913	-1.231378



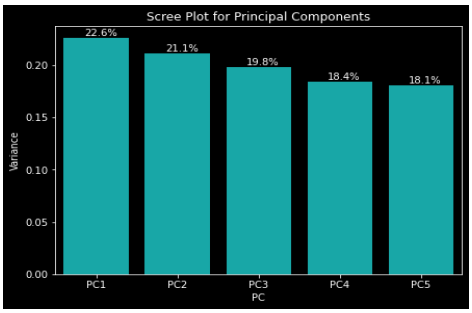
- The dataset is almost **balanced** as it has **41.5%** Fail values and **58.5%** Pass values.
- We notice that the distributions are **not normal** and as explained by the skewness values, the distributions are **approximately symmetric**.
  We split the data into train and test sets such that test data is 30% of the original data. We **standardise** the data such that the distribution has mean = 0 and variance = 1.



- **Linear discriminant analysis** (LDA) is a type of linear combination, a mathematical process using various data items and applying functions to that set to separately analyze multiple classes of objects or items. We perform Linear Discriminant analysis on the features such that we find a new axis which minimises intraclass variation and maximises interclass variation. If there are **'n' classes**, LDA returns **'n-1' axes**. In this case since we have only **2 classes**, LDA gives us only **1 axis**. On plotting, we see that the data projected onto this axis is quite separable, however there is some overlap among the classes.

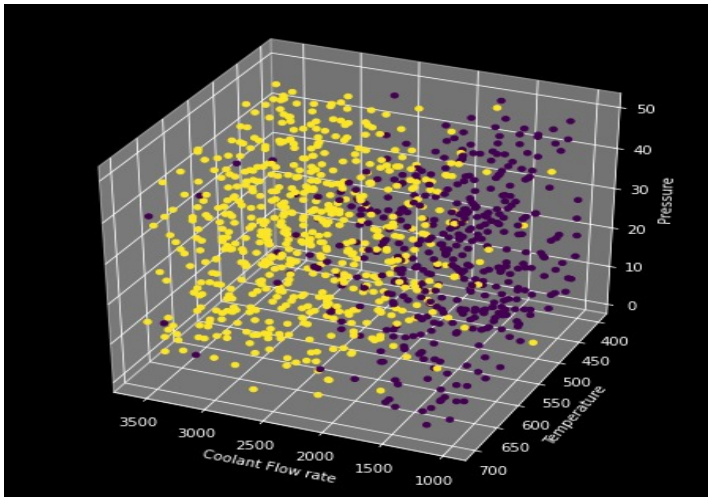


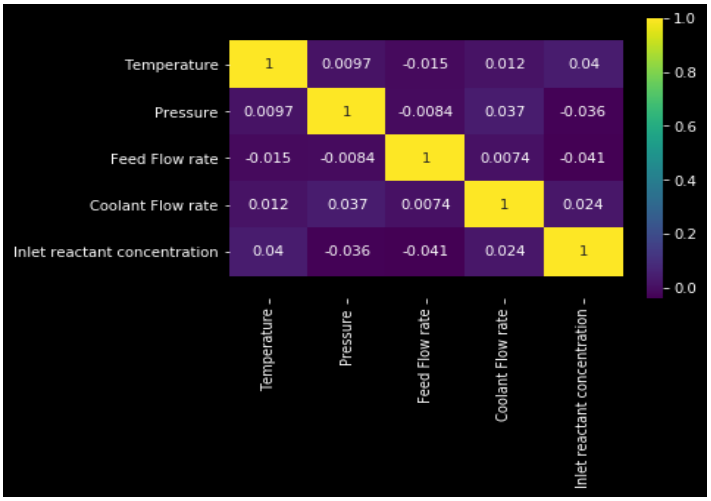
- **Principal component analysis** (PCA) is a technique used for identification of a smaller number of uncorrelated variables known as principal components from a larger set of data. On performing PCA, we get the same number of axes as the number of features in the original dataset however we check the **scree plot** for principal component axes with maximum variance and choose a few such that the loss in variance is not too high. However, in this scree plot we see that **all 5 Principal Component Axes have similar variances** and leaving out one would result in a **loss of about 18%**. Thus, applying PCA in this situation is not advisable.



Both LDA and PCA are linear transformation techniques; LDA is a supervised whereas PCA is unsupervised – PCA ignores class labels. We can picture PCA as a technique that finds the directions of maximal variance. In contrast to PCA, LDA attempts to find a feature subspace that maximizes class separability.

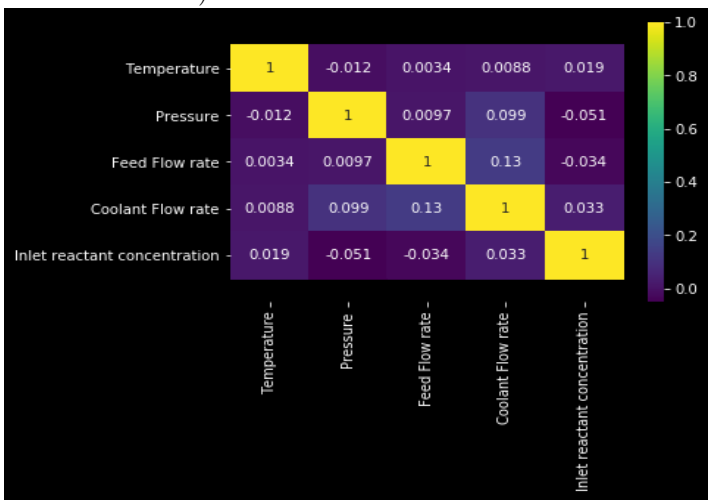
- Below, we have plotted three features; Coolant flow rate, Temperature and Pressure. We can see that there is a clear variation in the coolant flow rate axis.

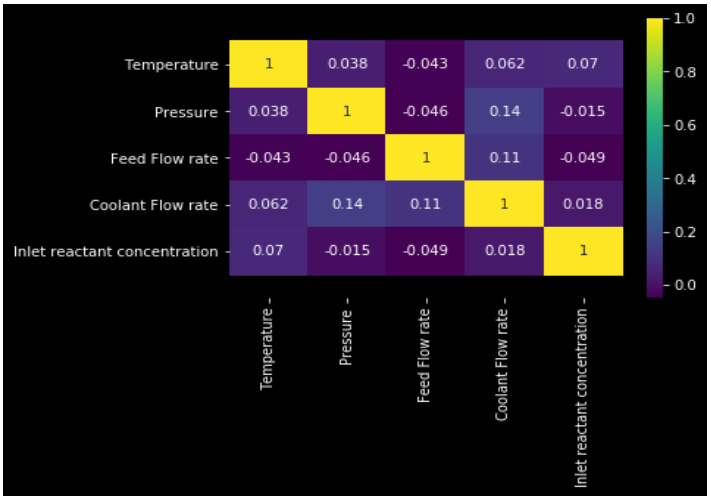




- **Total Correlation Matrix:** In the total correlation map, none of the features have correlation higher than **0.04**.

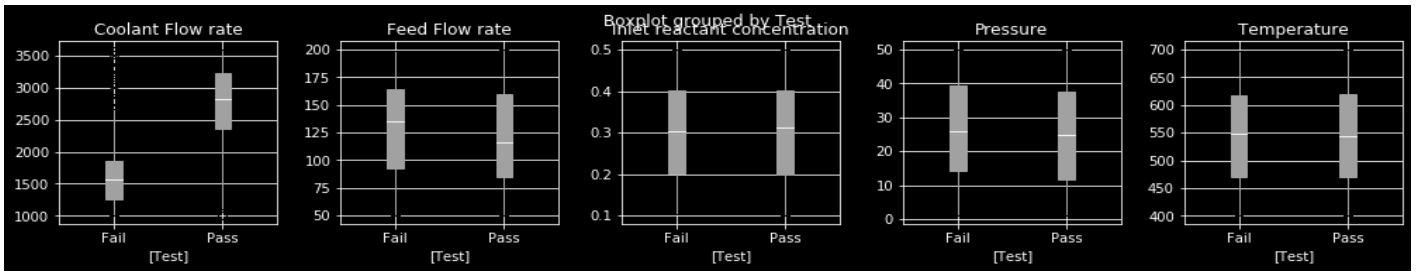
- **Classwise Correlation Matrix:** In the class-wise correlation map, the correlation is a little higher, **0.14** (pressure vs coolant flow rate).



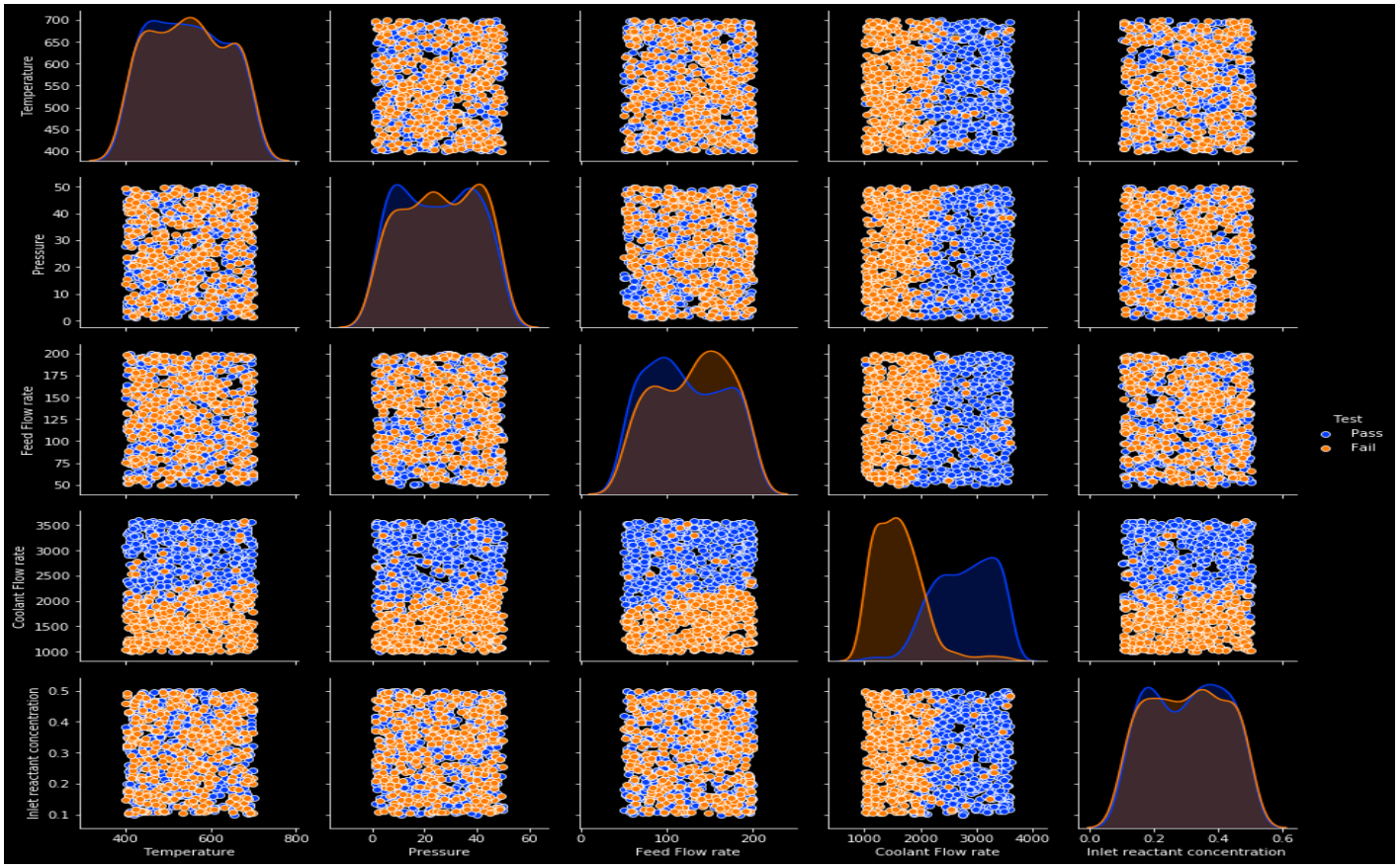


However, these correlations are extremely small and hence the features are considered to be uncorrelated.

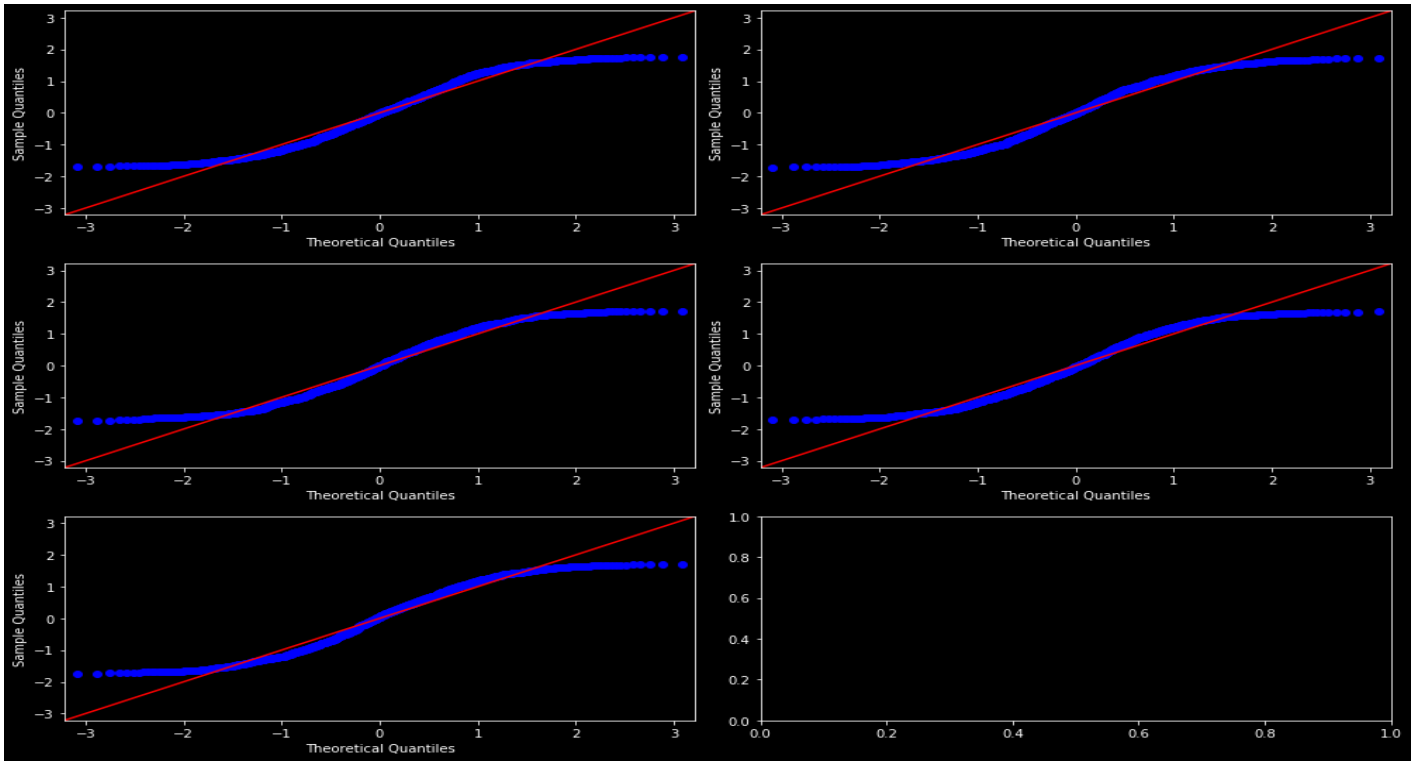
- **Classwise Difference in Features:** We can see that only **'Coolant flow rate'** is the only **well separated** among the classes, the other features have very similar range.
Inlet Reactant Concentration and Temperature have the same range; Feed flow rate and Pressure have little variation.



- **Bi-variate Plotting:** Most of the features show no separability except the features plotted with coolant flow rate.



Analysing distribution of features using Q-Q Plots



- As seen in the plots above, on a **Q-Q plot** under-dispersed data appears **S shaped**. Analysing the distribution of the features- none of the features come from normal distribution. The data appears to be **under-dispersed** with respect to a normal distribution. Under-dispersed data has a **reduced number of outliers** in comparison with the normal distribution. Under-dispersed data is also known as having a **platykurtic distribution** and as having negative excess kurtosis. All the features come from a similar distribution. They have been **standardised** to have zero mean and unit variance while their minimum, maximum and kurtosis values are similar as well. All the features have negative kurtosis. From the pairplot above, the distribution appears to be **multimodal** (a probability distribution with more than one peak, or mode).
- **Logistic Regression**: Logistic regression is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). Like all regression analyses, the logistic regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

  In the logistic model, $p(x)$ is interpreted as the probability of the dependent variable Y equaling a success/case rather than a failure/non-case. It is clear that the response variables Y_i are not identically distributed: $P(Y_i = 1 \setminus X)$ differs from one data point X_i to another, though they are independent given design matrix X and shared parameters β.
  We fit a binary logistic regression model on the training set, which predicts whether the reactor will pass or fail for the conditions provided by the test set. Sigmoid function is used to map predictions to probabilities, where the probabilty of pass or fail is given by,

$$\widehat{y} = \frac{1}{1+e^{-z}}$$

where z is the the multiple linear regression equation here given by

$$z = x.w^t + b$$

x being an Nx5 data matrix, b is the bias which is broadcasted to Nx1 and w is the weight matrix of size 1x5

Cross Entropy Loss

$L = -(ylog\widehat{y} + (1-y)log(1-\widehat{y}))$

$\frac{dL}{d\widehat{y}} = -\left(\frac{y}{\widehat{y}} - \frac{(1-y)}{(1-\widehat{y})}\right)$

$\frac{d\widehat{y}}{dz} = \widehat{y}(1-\widehat{y})$

$\frac{dL}{dz} = \frac{dL}{d\widehat{y}}\frac{d\widehat{y}}{dz}$

$\frac{dL}{dz} = -\left(\frac{y}{\widehat{y}}(\widehat{y})(1-\widehat{y}) - \left(\frac{1-y}{1-\widehat{y}}\right)(\widehat{y})(1-\widehat{y})\right)$

$= -(y-\widehat{y})$

$\frac{dz}{dw} = x$

$\frac{dz}{db} = 1$

$\frac{dL}{dw} = (\widehat{y}-y)x$

$\frac{dL}{db} = (\widehat{y}-y)$

Mean Squared Error

$L = \frac{1}{2}(y-\widehat{y})^2$

$\frac{dL}{d\widehat{y}} = -(y-\widehat{y})$

$\frac{d\widehat{y}}{dz} = \widehat{y}(1-\widehat{y})$

$\frac{dz}{dw} = x$

$\frac{dz}{db} = 1$

$\frac{dL}{dw} = (y-\widehat{y})(\widehat{y})(1-\widehat{y})x$

$\frac{dL}{db} = (\widehat{y}-y)y(1-\widehat{y})$

```python
def logistic_regression(x,w,b):
    '''x -> data_matrix Nx5
       w -> weight_matrix 1x5
       b -> bias 1x1  broadcasted to Nx1
       output -> sigmoid(x.wT+b) size : Nx1'''
    return 1/(1 + np.exp(-(np.matmul(x,w.T)+b)))

def grad_w(x,ypred,y):
    return np.matmul((ypred-y).T,x)

def grad_b(ypred,y):
    return (ypred - y).mean(axis=0)

def grad_w_mse(x,ypred,y):
    u = (ypred - y)*ypred*(1-ypred)
    #print(u.shape)
    return np.matmul(u.T,x)

def grad_b_mse(ypred,y):
    return ((ypred - y)*ypred*(1-ypred)).mean(axis=0)

def transform(ypred):
    ypred[ypred>=0.5]=1
    ypred[ypred<0.5]=0
    return(ypred)
```

```python
def fit(xtrain,ytrain,xval=xtest,yval=ytest,epochs=1000,lr=0.01):
    loss_ar=[]
    weights = np.random.randn(1,xtrain.shape[1])
    bias = 0
    test_acc_best=0
    train_acc_best=0
    for epoch in range(epochs):

        ypred_train = logistic_regression(xtrain,weights,bias)
        ypred_val = logistic_regression(xval,weights,bias)
        ypred_train_t = transform(ypred_train)
        ypred_val_t = transform(ypred_val)

        weights-=lr*grad_w(xtrain,ypred_train,ytrain)
        bias-=lr*grad_b(ypred_train,ytrain)


        loss = mean_squared_error(ytrain,ypred_train)
        loss_ar.append(loss)
        train_acc=accuracy_score(ytrain,ypred_train_t)
        test_acc=accuracy_score(yval,ypred_val_t)

        if train_acc>train_acc_best and test_acc>test_acc_best:
            train_acc_best=train_acc.copy()
            test_acc_best=test_acc.copy()
            best_weights=weights
            best_bias=bias
        #print('Epoch {}/{} : Train_accuracy {:.2f}, Validation Accuracy {:.2f}'.format(epoch+1, epochs,train_acc ,
    return (loss_ar,best_weights,best_bias,train_acc_best,test_acc_best)
```

Using Cross Entropy Loss

The most appropriate objective function for classification using logistic regression is the cross entropy loss function because minimizing this is equivalent to obtaining the maximum likelihood estimate for the weight. Intuitively this minimizes the distance between two probability distributions predicted and actual.

- Building model using the **Original Data**.

```
## Original Data
loss_ar,best_weights,best_bias,train_acc_best,test_acc_best=fit(xtrain,ytrain,epochs=1000,lr=0.001)
```

```
ypred_orig = logistic_regression(xtest,w=best_weights, b=best_bias)
predictions_orig = [np.round(value) for value in ypred_orig]
f1_score_orig = f1_score(ytest,predictions_orig)
cm_orig = confusion_matrix(predictions_orig,ytest)
print('f1 score',f1_score_orig)
print('Using original dataset best test accuracy is {:.2f}%'.format(100*test_acc_best))
print('Confusion Matrix',cm_orig,sep='\n')
```

```
f1 score 0.819672131147541
Using original dataset best test accuracy is 92.67%
Confusion Matrix
[[120  50]
 [  5 125]]
```

- Building model using **LDA Data**.

```
## LDA Data
loss_ar,best_weights_lda,best_bias_lda,train_acc_best,test_acc_best=fit(xtrain_lda,ytrain,xval=xtest_lda,epochs=200
```

```
ypred_lda = logistic_regression(xtest_lda,w=best_weights_lda, b=best_bias_lda)
predictions_lda = [np.round(value) for value in ypred_lda]
f1_score_lda = f1_score(ytest,predictions_lda)
cm_lda = confusion_matrix(predictions_lda,ytest)
print('f1 score',f1_score_lda)
print('Using LDA dataset best test accuracy is {:.2f}%'.format(100*test_acc_best))
print('Confusion Matrix',cm_lda,sep='\n')
```

```
f1 score 0.9194029850746268
Using LDA dataset best test accuracy is 93.33%
Confusion Matrix
[[119  21]
 [  6 154]]
```

- Building model using the **Coolant Flow Rate Data**.

```
## Coolant Flow Rate
xtrain_cool=np.expand_dims(xtrain[:,3],axis=1)
xtest_cool=np.expand_dims(xtest[:,3],axis=1)
loss_ar,best_weights_cool,best_bias_cool,train_acc_best,test_acc_best=fit(xtrain_cool,ytrain,xval=xtest_cool,epochs
```

```
ypred_cool = logistic_regression(xtest_cool,w=best_weights_cool, b=best_bias_cool)
predictions_cool = [np.round(value) for value in ypred_cool]
cm_cool = confusion_matrix(predictions_cool,ytest)
f1_score_cool = f1_score(ytest,predictions_cool)
print('f1 score',f1_score_cool)
print('Using Coolant flow rate best test accuracy is {:.2f}%'.format(100*test_acc_best))
print('Confusion Matrix',cm_cool,sep='\n')
```

```
f1 score 0.8900523560209423
Using Coolant flow rate best test accuracy is 91.00%
Confusion Matrix
[[ 88   5]
 [ 37 170]]
```

Here, coolant flow rate alone was chosen to build a model since it displayed clear variation in the distribution for the two classes, we can see that the accuracy is comparable to the one with entire data.

Using Mean Squared Error

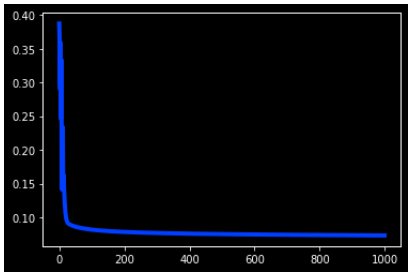
Th mean squared error is more suitable in cases where the response variable is continuous. However in this case, our target is discrete since we perform classification.

- Building model using the **Original Data**.

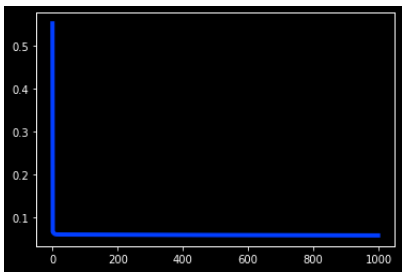
```
epochs = 1000
lr=0.09
x=xtrain
y=ytrain
w=np.random.randn(1,x.shape[1])
b=np.asarray([1.5])
loss_ar=[]
for i in range(epochs):
    ypred=logistic_regression(x,w,b)
    w-=grad_w_mse(x,ypred,y)*lr
    b-=grad_b_mse(ypred,y)*lr
    loss_ar.append(mean_squared_error(ypred,y))
```

```
ypred_orig = logistic_regression(xtest,w,b)
pred_orig = transform(ypred_orig)
accuracy_score(ytest,pred_orig)
```

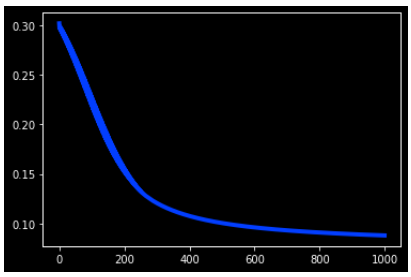
Original data



LDA data

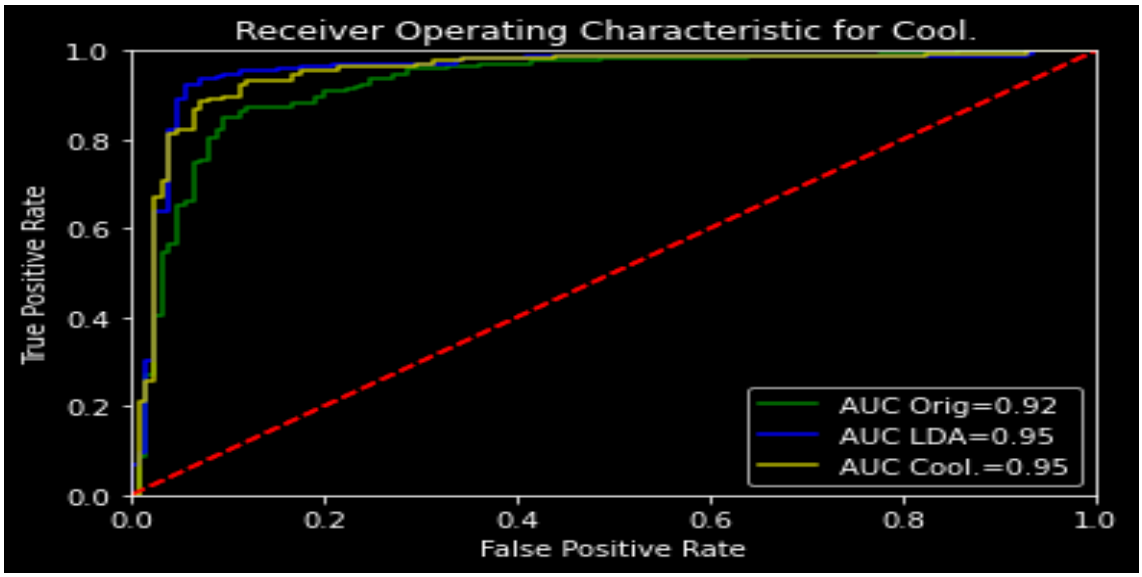


Coolant flow rate

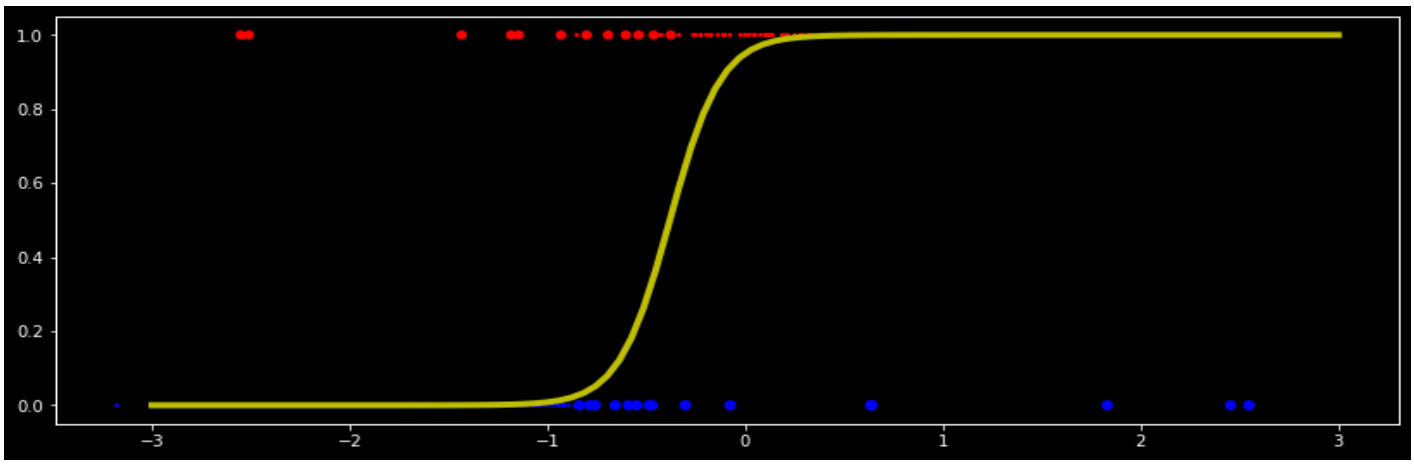


	Cross Entropy Loss	**Mean Squared Error**
ORIGINAL DATA	– Accuracy: 93.33% – F1 score 0.82 – Confusion matrix $\begin{bmatrix} 120 & 50 \\ 5 & 125 \end{bmatrix}$	– Accuracy: 92% – F1 score 0.93 – Confusion matrix $\begin{bmatrix} 111 & 14 \\ 10 & 165 \end{bmatrix}$
LDA DATA	– Accuracy: 93.33% – F1 score 0.92 – Confusion matrix $\begin{bmatrix} 119 & 21 \\ 6 & 154 \end{bmatrix}$	– Accuracy: 92.7% – F1 score 0.94 – Confusion matrix $\begin{bmatrix} 117 & 8 \\ 13 & 162 \end{bmatrix}$
COOLANT FLOW RATE DATA	– Accuracy: 91% – F1 score 0.89 – Confusion matrix $\begin{bmatrix} 88 & 5 \\ 37 & 170 \end{bmatrix}$	– Accuracy: 91% – F1 score 0.92 – Confusion matrix $\begin{bmatrix} 110 & 15 \\ 12 & 163 \end{bmatrix}$

- From the **ROC-AUC** curve we can conclude that the LDA model and Coolant Flow Rate model are better than the model trained on the original data. Combining accuracy, F1 score and area under curve (AUC), we can conclude that the model trained on LDA data performs the best.



- In these plots, the correctly classified points are small in size, while the incorrectly classified points are the large ones. From this we can get a rough idea of the decision boundary. Since the data is 5 dimensional, it cannot be plotted directly, thus we use the LDA data to visualise the test set.

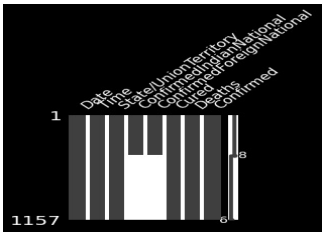


The model visualised above is the **LDA data** model.

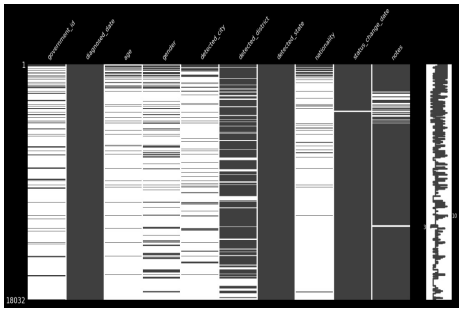
Question 3

Original dataset had dates till 4-4-20 however the question demanded quantification till 10-4-20. Hence the dataset was downloaded from kaggle using the link provided, `https://www.kaggle.com/sudalairajkumar/covid19-in-india/discussion/143911`. However in this updated dataset, the status change column was removed. This is due to the inaccuracy of the values reported in the column. The link provided validates the claim.

Imputation Process



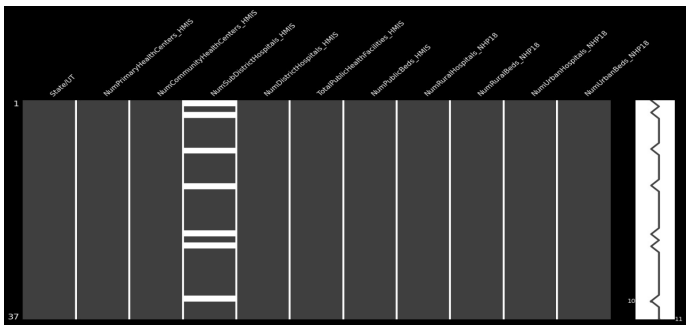
covid_19_india.csv
We can see that 'ConfirmedIndianNational', 'ConfirmedForeignNational' have many missing values. Assuming that there has been no entry of foreign nationals after the last available value, we find number of indian cases as total confirmed cases - foreign cases.



Individual_cases.csv
There are no missing detected_state values. For each state, there are some districts mentioned and others are null values. The districts are filled in according to the existing distribution of districts for each state. Gender and Age are also filled this way. In this method, the data is grouped according to detected_state first; each grouping has a distribution of detected_district; the missing values are filled by sampling values according to the probability distribution of districts for each state. Once detected_state is filled, we grouped by detected_state and filled detected_city in the same way. We sample age and gender similarly, using their respective district wise distributions.

Even after doing this, Arunachal Pradesh had a district with only one case with no age, gender. This was imputed with the mode of the overall data. Even after this, there were districts that had no cities filled in them. For these cases, the city was set as the district. Nationality was assumed to be Indian for those cases where not mentioned. Variations of the same nationality were standardised. Eg, Indian to India. Notes that were not mentioned have been set as 'Details Awaited'.

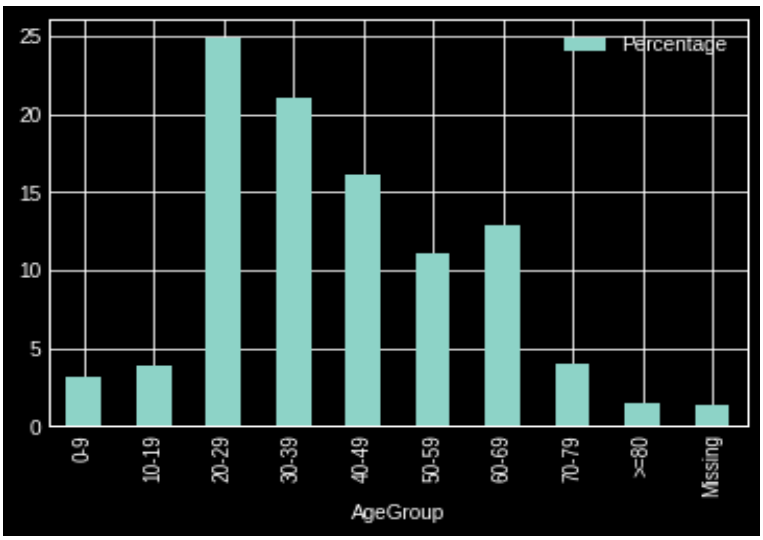
The samples are chosen randomly but according to their probability distribution. For a particular missing value it is random but the overall distribution of the feature remains.



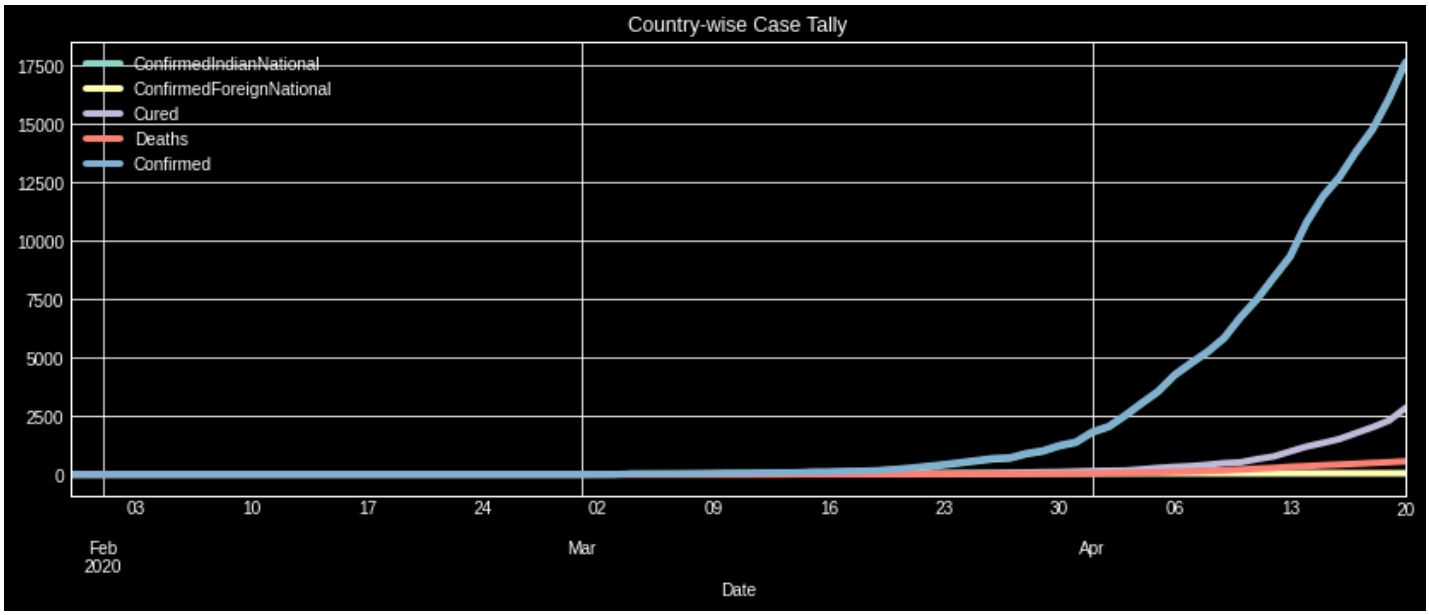
HospitalBedsIndia.csv
NumSubDistrictHospitals_HMIS has 19% missing values, however, on adding the remaining values, it equals the last row **'All India'** value. Thus, the NaN values have been replaced with zero.

The other datasets have no missing values.

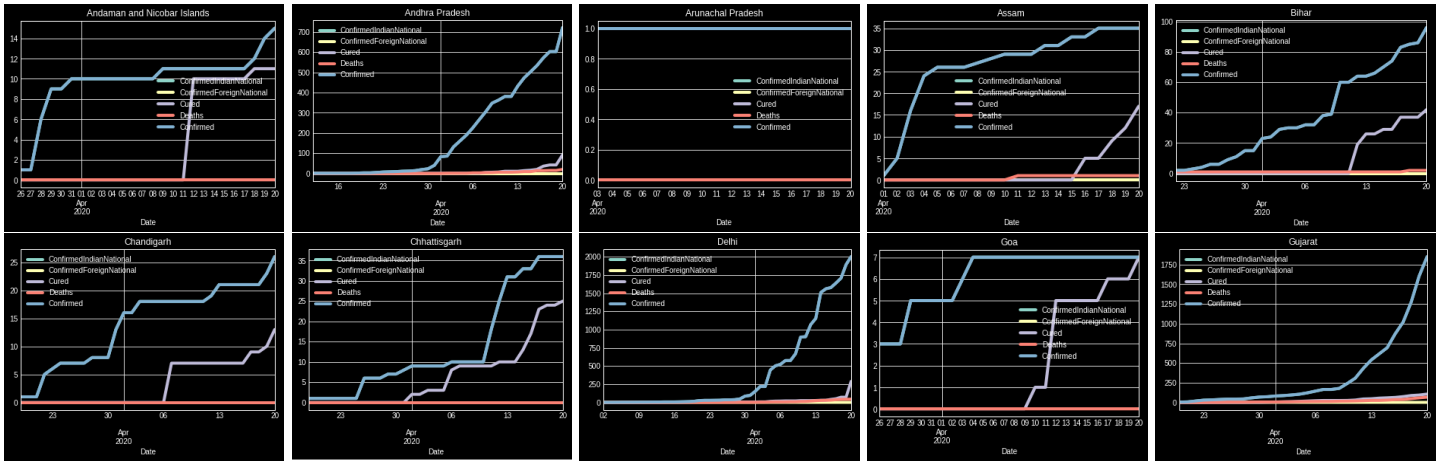
1. From the plot below, we can tell that the maximum number of cases has occurred in the age group of **20-29** that is **25% of the total**.
   The missing category was not imputed because the margin between the most frequently occurring age group and the second most frequently occurring age group is more than the percentage of values in the missing category. So we can conclude that the the maximum number of cases have occurred in the age group of 20-29 irrespective of accounting the missing data.

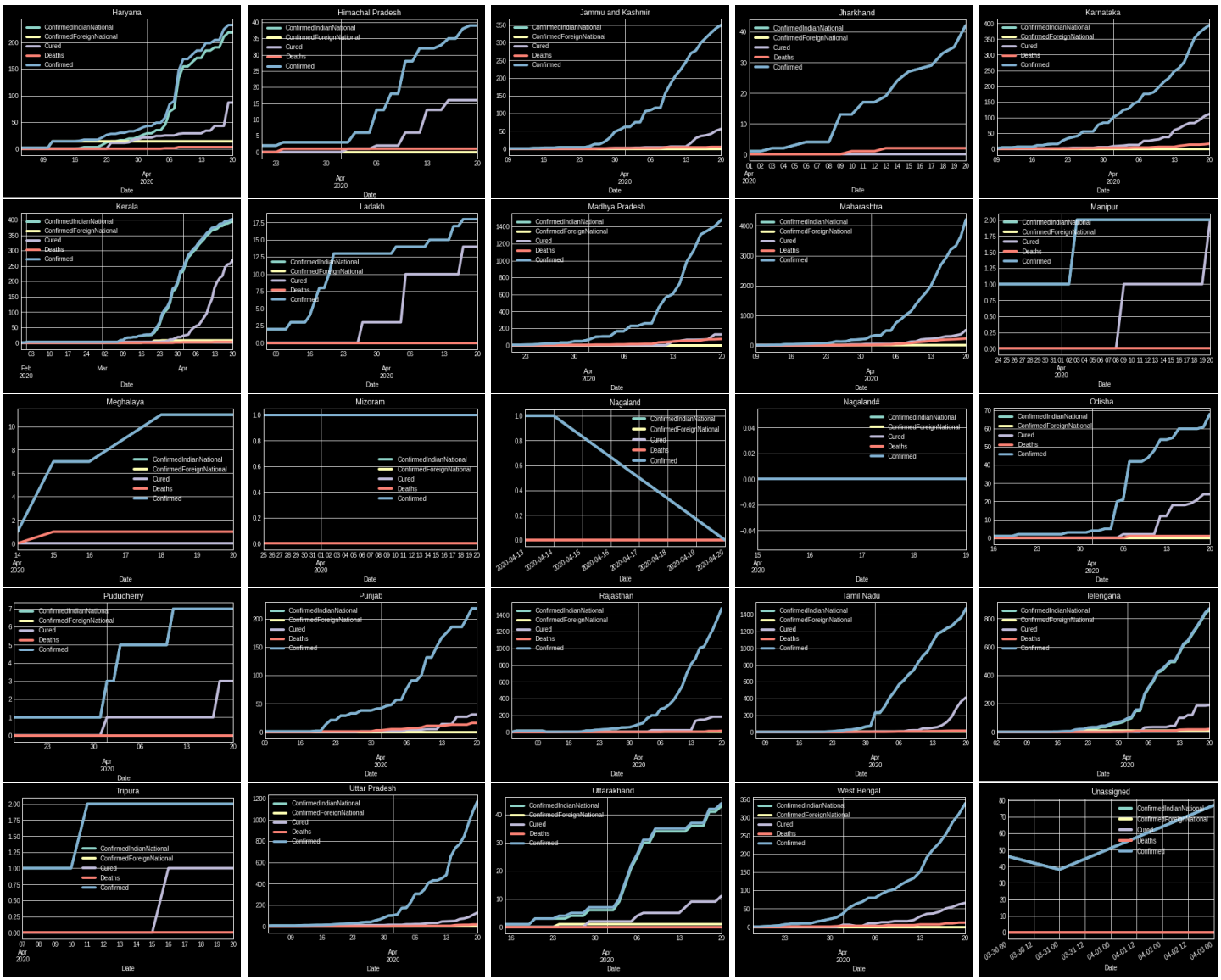


2. Graphs representing **Country-wise cases observed, recovered, deaths**.

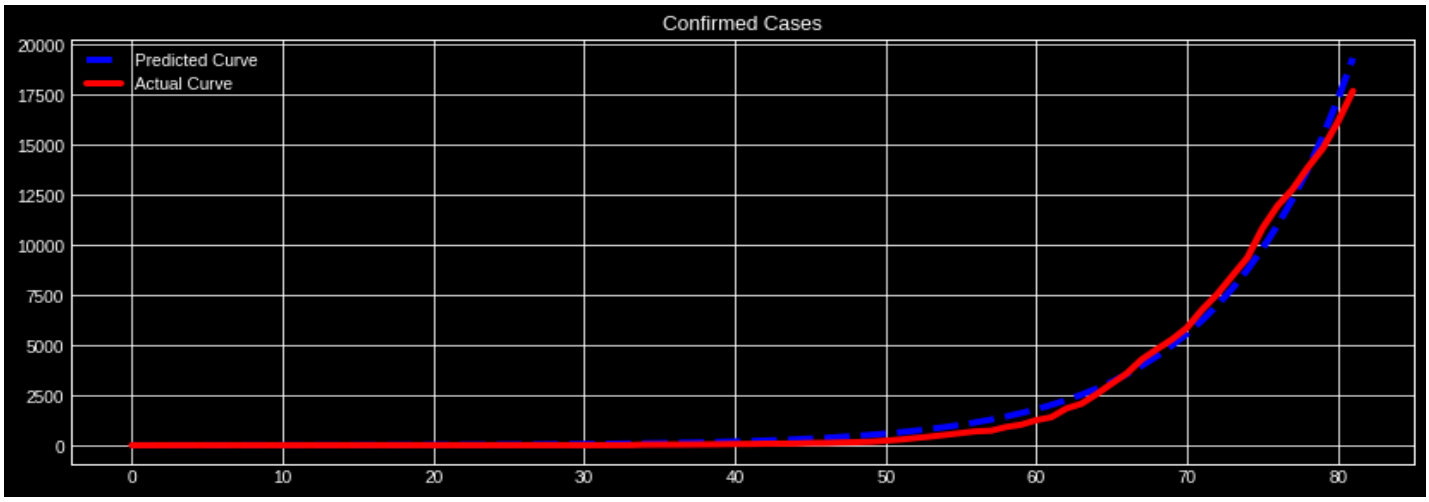


Graphs representing **State-wise cases observed, recovered, deaths**.

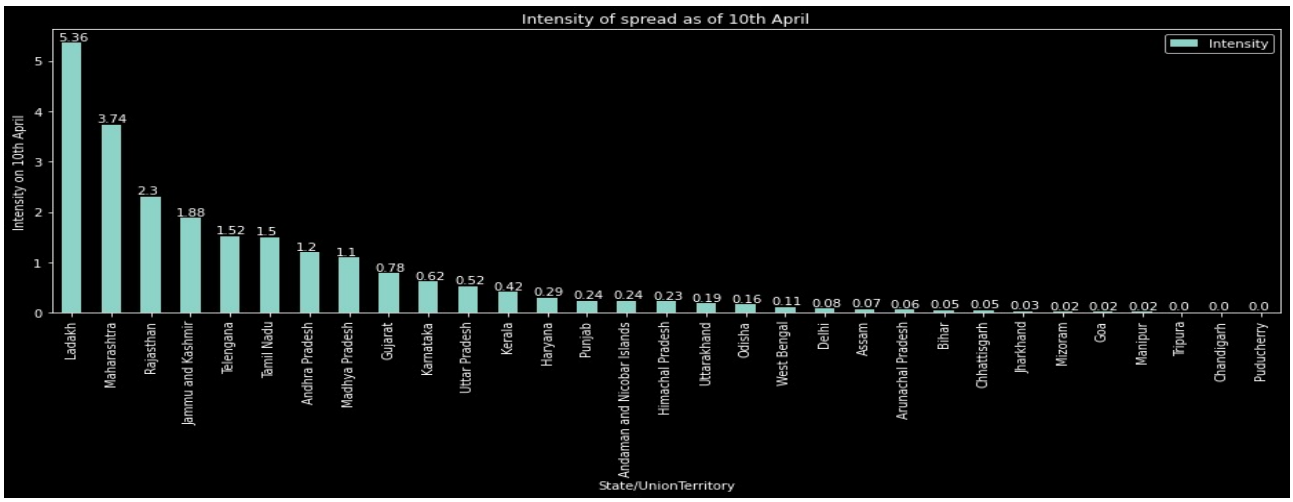




On performing **Polynomial Curve Fitting** for the country wise total number of confirmed cases, we get $2.04e^{0.11x}$ to be the curve. R^2 score of the fit is **0.99**.



3. Intensity of the states as of 10th April has been calculated.
Maharashtra has the maximum number of cases, closely followed by Delhi.



4. Places in the country which are active hotspots/clusters as on 10.04.2020 are listed below. There are **172** hotspots in total across the country.
   MCGM, Nizamuddin area, Mumbai, Indore, Delhi, Kasaragod, Hyderabad, Ahmedabad, Ramganj, Bhopal, Agra, Chennai, Jaisalmer, PMC, Jaipur, Mettupalayam, Bengaluru, Pimpri-Chinchwad, Pune, Vadodara, Begampur, Perundurai, Jahangirpuri, Nizamabad, Ahmadabad, Janakpuri, Gurugram, Meerut, Navi Mumbai, Namakkal, Hajin, Nuh, Tiruchirappalli, Walajapet, Kalyan-Dombivali, Melapalayam, Noida, Mumbai Suburb, Jhunjhunu, Vijayawada, Nanjangud, Mohali, Siwan, Kadappa, Kashmir Division, Lucknow, Bodi, Purasaivakkam, Palwal, Faridabad, Thane, Tonk, Viluppuram, Surat, Bhilwara, Sangli, Tiruppur, Bhubaneswar, Rana Pratap Bagh, Madurai, Nagpur, Tirunelveli, Thoothukkudi, Shashtri Nagar, Banswara, Kochi, Karur, Ongole, Bhavnagar, Pune Rural, Dilshad Garden, Shastrinagar, Saket, Srinagar, Mira-Bhayandar, Nadia, Gautam Puri, Bikaner, Warangal Urban, Saharanpur, Kukatpally, Udhampur, Kota, Jogulamba Gadwal, Jodhpur, Mooriyad, Karimnagar, Kolkata, Dehradun, Ahmednagar, Nawanshahr, Rajkot, Domalguda, Chandigarh, Shamli, Natipora, Mumbai City, Ashok Vihar, West Delhi, Ghaziabad, Kurnool, Aurangabad, Keelkattalai, Nirmal, Tirupathur, Ujjain, Uttam Nagar, Annaimalai, Kanniyakumari, Kupwara, Sujjanpur, North Delhi, Jawaharpur, Broadway, Khargone, Gandhinagar, East Delhi (Mayur Vihar), Patan, Cuddalore, Salem, Barwani, Beleghata, Tri Nagar, Shupiyan, Nellore, Ariyalur, Thiruvallur, Kokapet, Anandpet, Thrissur, Saidabad, Morena, Buldana, Thiruvarur, Malappuram, Porur, Vasai-Virar, Nalgonda, Nagapattinam, Jhalawar, Jammu, Habra, Medinipur East, Anna Nagar, Akola, PCMC, Phillaur, Una, Amritsar, Kumbakonam, Kamareddy, Badgam, Mahabubnagar, Chirala, Rajapalayam, Kaushambi, Pammal, Churu, Kathipudi, Katpadi, Firozabad, Mansa, Bombooflat, Rajouri, Maradu, Adilabad, Penugonda, Ludhiana, Chandanagar, Kalaburagi, Somajiguda, Sitapur, Katghora.

5. The given time period of three weeks comprises of the following dates.
   - 20th March to 26th March is considered to be Week 1.
   - 27th March to 3rd April is considered to be Week 2.
   - 4th April to 10th April is considered to be Week 3.

Number of hotspots state-wise by the end of Week 1

Maharashtra	3
Kerala	2
Gujarat	1
Punjab	1
Rajasthan	1
Uttar Pradesh	1
Karnataka	1
Tamil Nadu	1
Haryana	1
Delhi	1
Telangana	1

Number of hotspots state-wise by the end of Week 2

Tamil Nadu	15
Maharashtra	10
Delhi	6
Uttar Pradesh	5
Rajasthan	5
Telangana	5
Gujarat	5
Kerala	4
Jammu and Kashmir	3
Karnataka	2
Andhra Pradesh	2
Madhya Pradesh	2
Punjab	2
Uttarakhand	1
Haryana	1
West Bengal	1
Chandigarh	1

Number of hotspots state-wise by the end of Week 3

Tamil Nadu	33
Maharashtra	19
Rajasthan	15
Delhi	14
Telangana	14
Jammu and Kashmir	10
Uttar Pradesh	10
Gujarat	8
Andhra Pradesh	8
Punjab	8
Kerala	7
Madhya Pradesh	6
West Bengal	4
Haryana	4
Karnataka	4
Uttarakhand	1
Odisha	1
Chhattisgarh	1
Himachal Pradesh	1
Chandigarh	1
Bihar	1
Andaman and Nicobar Islands	1

Since there is randomness in the data imputed, the number of hotspots keep changing.

Changes in number of hotspots from Week 1 to Week 2.

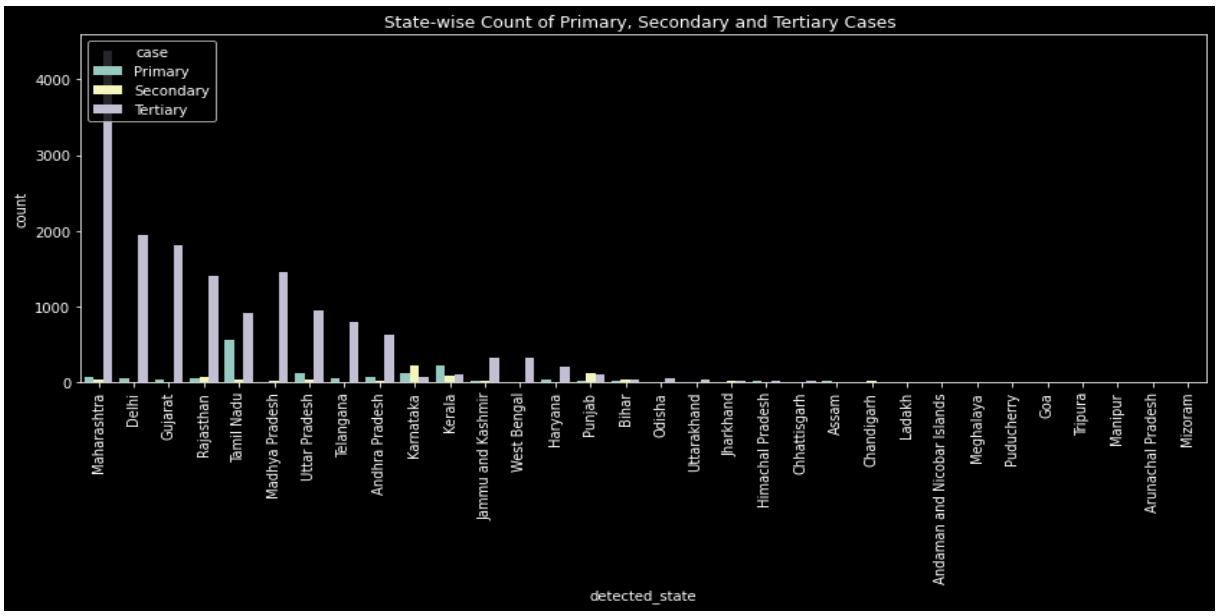
	detected_state
Tamil Nadu	14.0
Jammu and Kashmir	10.0
Andhra Pradesh	8.0
Maharashtra	7.0
Madhya Pradesh	6.0
Delhi	5.0
Gujarat	4.0
Rajasthan	4.0
Telangana	4.0
Uttar Pradesh	4.0
West Bengal	4.0
Kerala	2.0
Karnataka	1.0
Chandigarh	1.0
Punjab	1.0
Uttarakhand	1.0
Haryana	0.0

Changes in number of hotspots from Week 2 to Week 3.

	detected_state
Tamil Nadu	18.0
Rajasthan	10.0
Telangana	9.0
Maharashtra	9.0
Delhi	8.0
Jammu and Kashmir	7.0
Andhra Pradesh	6.0
Punjab	6.0
Uttar Pradesh	5.0
Madhya Pradesh	4.0
Kerala	3.0
West Bengal	3.0
Haryana	3.0
Gujarat	3.0
Karnataka	2.0
Himachal Pradesh	1.0
Odisha	1.0
Chhattisgarh	1.0
Bihar	1.0
Andaman and Nicobar Islands	1.0
Chandigarh	0.0
Uttarakhand	0.0

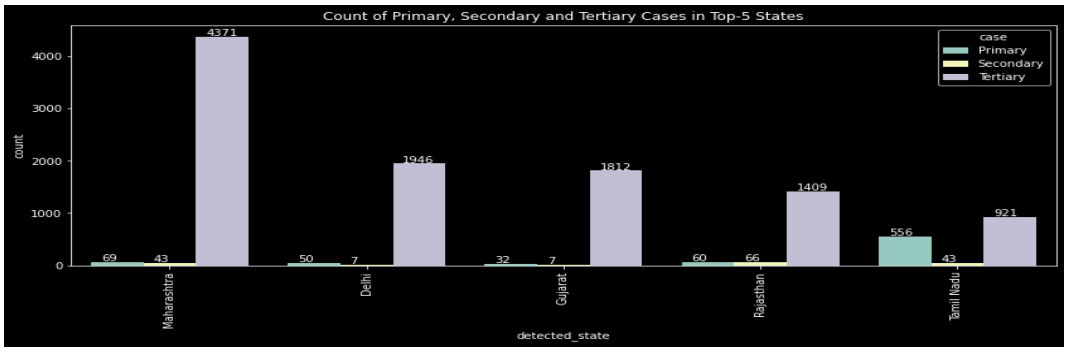
6. In the individaual cases data set, there is a column called 'notes' which mentions if there is any travel history or contact history. Most of the cases have no details mentioned and these have been assumed as Tertiary cases. On studying the notes, it can be seen that the notes with the terms **travel**, **airport** but not containing the words **no** and **sibling** are the one's with travel history; and the notes containing **contact**, **relative**, **related**, **family**, **conference**, **Nizamuddin**, **patient number** but not containing **second**, **travel**, **local** and **no** are considered to be secondary cases.

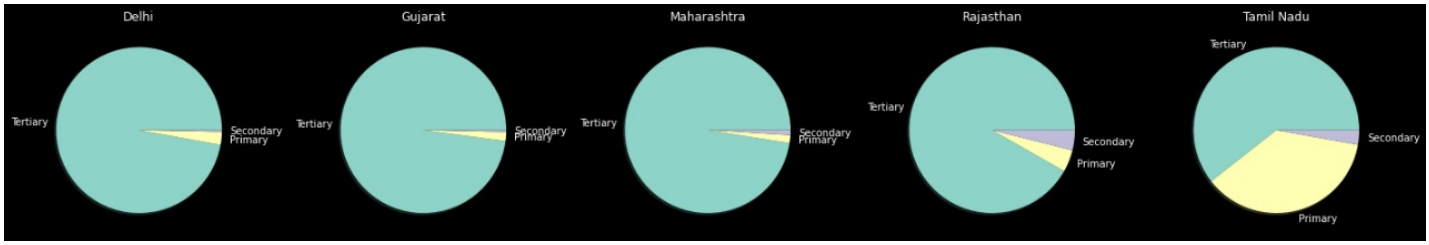
	Primary	**Secondary**	**Tertiary**
Number of cases	1567	793	15672
Percentage of the total cases	8.7%	4.4%	86.9%



Top 5 States with maximum number of total cases

Maharashtra	4483
Delhi	2003
Gujarat	1851
Rajasthan	1535
Tamil Nadu	1520





7. Number of cases on 10th April = 6761

With an increase rate of 10% cases per day, number of cases on 20th April = $6761\left(1+(\frac{10}{100})^{10}\right) = 17537$

Therefore, number of additional Cases from 11th to 20th April = 17537 - 6761 = 10776

	DateTime	TotalSamplesTested	TotalIndividualsTested	TotalPositiveCases	Positive Rate
0	2020-03-13	6500	5900	78	0.012000
5	2020-03-18	13125	12235	150	0.011429
6	2020-03-19	27491	25711	350	0.012731
7	2020-03-20	29780	28000	442	0.014842
8	2020-03-21	32612	30832	586	0.017969
9	2020-03-22	35126	33346	737	0.020982
10	2020-03-23	39090	37310	886	0.022666
11	2020-03-24	43558	41778	1018	0.023371
12	2020-03-25	48072	46292	1120	0.023298

From the table,
Last available positive rate = 0.023298
Thus, for every 1 sample tested, 0.023298 are positive.
Thus, to find 1 positive sample, the number of samples to be tested = $\left(\frac{1}{0.023298}\right)$
Therefore, to find 10776 positive samples, the number samples to be tested = $\left(\frac{1}{0.023298}\right)10776 = 462529$

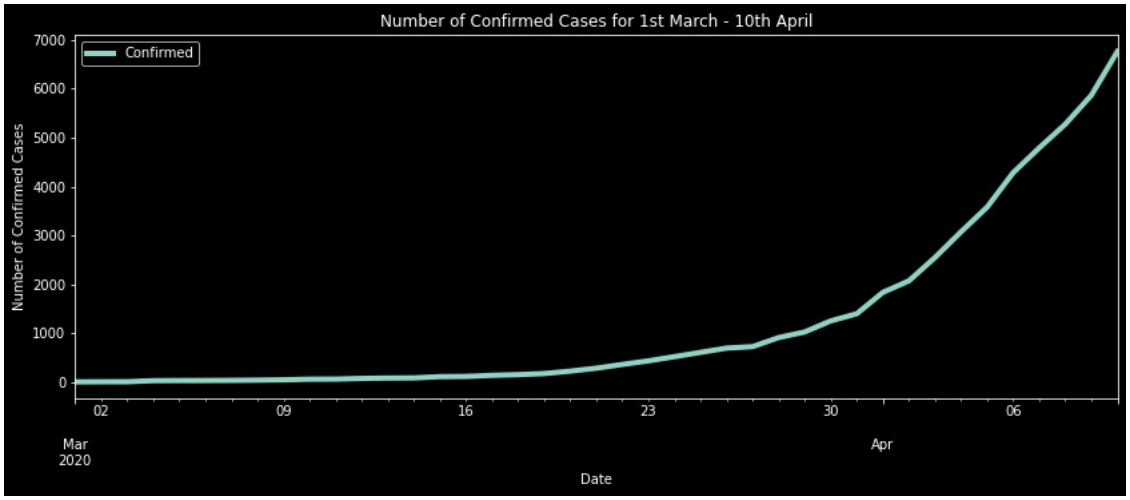
Number of tests done by 1 lab in 1 day = 100
Number of days from 11th to 20th April = 10
Number of tests done by 1 lab in 10 days = 1000
Total number of labs required = $\left(\frac{462529}{1000}\right)$
Number of existing labs = $\left(\frac{48072}{13\times100}\right) = 212$
Therefore, additional number of labs required = 463 - 212 = **251**

Assumptions made:
Since testing data is available only from 13th March - 25th March, we assume

- The proportion of positive samples does not change from 25th March to 20th April.
- The number of labs performing tests from 13th March to 10th April does not change, all new labs are built after 10th April.

8. Notion of **"Flattening the Curve"** -
The curve shows the cumulative count of people affected by the COVID-19 in India between 1st March and 10th April. Flattening the curve refers to the degree of community isolation so that the number of cases are less than health care capacity of the country.
If the infection rate of virus is exponential the curve would be a steep curve with number of cases reaching it's peak within a short time. Infection curve with steep rise will also have steep fall as virus would have infected mostly everyone who can be infected. So, the number of cases drops exponentially.
These will overload the health care system and the new patients may be forced to go without ventilators and other critical care equipment(as we have seen in Italy). If Community Isolation/Social Distancing is followed strictly the number of people

getting infected will be same but over a longer period of time. These will lead to a flatter curve and will not overload the health care system.



9. Based on the time series data from the file **covid_19_india.csv**, we use the criterion of weekly growth multiple to measure how effective the 21 day lockdown has been.

$$Weekly\ growth\ multiple = \frac{Cumulative\ cases\ at\ the\ end\ of\ present\ week}{Cumulative\ cases\ end\ of\ previous\ week}$$

Based on the data, the 21 day lockdown began on the 24th of March
Number of Cases as of 17th March = 137

Week	**Number of Cases**	**Weekly growth multiple**
25th March	606	$\frac{606}{137} = 4.42$
1st April	1966	$\frac{1966}{606} = 3.24$
8th April	5749	$\frac{5749}{1966} = 2.92$
15th April	12021	$\frac{12021}{5749} = 2.09$
22nd April	20004	$\frac{20004}{12021} = 1.66$

As the figures clearly indicate that social distancing followed in the 21 day lockdown was effective. If it would not have been followed, then assuming the same weekly growth multiple as before the number of cases by end of lockdown, as of 15th April would be approximately 52,448 but we have 20,004 cases. We were able to avoid 32,444 extra cases, with our strict social distancing measures.
The above figures give us ample evidence that the 21 day lockdown was **highly successful** as we could avoid **61.86%** of the expected number of cases without lockdown.

References

[1] Chou-Hao Hsu and Chaur-Chin Chen. "SVD-Based Projection for Face Recognition". In: *IEEE International Conference on Electro/Information Technology* (2007).

[2] Machine Learning- LDA and PCA for dimensionality reduction. URL: `https://sebastianraschka.com/faq/docs/lda-vs-pca.html`.

[3] Examples of Quantile-Quantile Plots by Dr Jon Yearsley. URL: `http://www.ucd.ie/ecomodel/Resources/QQplots_WebVersion.html`.

[4] News 18 India- Success of lockdown in preventing the spread of novel coronavirus. URL: `https://www.news18.com/news/india/month-after-junta-curfew-a-look-at-how-successful-india-has-been-post-worlds-most-stringent-covid-19-lockdown-2589017.html`.