**Programming for Analytics**

**Mini Project**

**DESCRIPTIVE STATISTICS, ML ALGORITHM, SQL QUERIES**

**FOR COVID19 DATASET**

**Submitted to**

**Dr Gajapathy V**

**Associate Professor**

**Prepared by**

**Pranil Raj**

**20182MBA0437**

**Presidency University**



**SCHOOL OF MANAGEMENT PRESIDENCY UNIVERSITY, BENGALURU – 560 064**

Descriptive analysis and ML model is done using google Collab.

!pip install squarify

!pip install plotly\_express

By default, google Collab have some library installed, but for this case we need to install squarify and plotly\_express, (helps for visualization that split the area of our chart to display the value of our datapoints)

**import** **pandas** **as** **pd**

**import** **numpy** **as** **np**

**import** **datetime**

**import** **requests**

**import** **warnings**

**import** **matplotlib.pyplot** **as** **plt**

**import** **matplotlib**

**import** **matplotlib.dates** **as** **mdates**

**import** **seaborn** **as** **sns**

**import** **squarify**

**import** **plotly.offline** **as** **py**

**import** **plotly\_express** **as** **px**

**from** **xgboost** **import** XGBRegressor

**from** **lightgbm** **import** LGBMRegressor

**from** **sklearn.ensemble** **import** RandomForestRegressor

**from** **sklearn.svm** **import** SVR

**from** **sklearn.metrics** **import** mean\_squared\_error, mean\_absolute\_error

**from** **sklearn.preprocessing** **import** OrdinalEncoder

**from** **sklearn.model\_selection** **import** train\_test\_split

**from** **statsmodels.tsa.arima\_model** **import** ARIMA

**from** **fbprophet** **import** Prophet

**from** **fbprophet.plot** **import** plot\_plotly, add\_changepoints\_to\_plot

**from** **IPython.display** **import** Image

warnings.filterwarnings('ignore')

%matplotlib inline

These are the library we need to import for this mini project

age\_details = pd.read\_csv('age\_details.csv')

india\_covid\_19 = pd.read\_csv('india\_covid\_19.csv')

hospital\_beds = pd.read\_csv('hospital\_beds.csv')

individual\_details = pd.read\_csv('individual\_details.csv')

ICMR\_details = pd.read\_csv('ICMR\_details.csv')

ICMR\_labs = pd.read\_csv('ICMR\_labs.csv')

state\_testing = pd.read\_csv('state\_testing.csv')

population = pd.read\_csv('population.csv')

world\_population = pd.read\_csv('world\_population.csv')

confirmed\_df = pd.read\_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse\_covid\_19\_data/csse\_covid\_19\_time\_series/time\_series\_covid19\_confirmed\_global.csv')

deaths\_df = pd.read\_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse\_covid\_19\_data/csse\_covid\_19\_time\_series/time\_series\_covid19\_deaths\_global.csv')

recovered\_df = pd.read\_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse\_covid\_19\_data/csse\_covid\_19\_time\_series/time\_series\_covid19\_recovered\_global.csv')

latest\_data = pd.read\_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse\_covid\_19\_data/csse\_covid\_19\_daily\_reports/04-04-2020.csv')

india\_covid\_19['Date'] = pd.to\_datetime(india\_covid\_19['Date'],dayfirst = **True**)

state\_testing['Date'] = pd.to\_datetime(state\_testing['Date'])

ICMR\_details['DateTime'] = pd.to\_datetime(ICMR\_details['DateTime'],dayfirst = **True**)

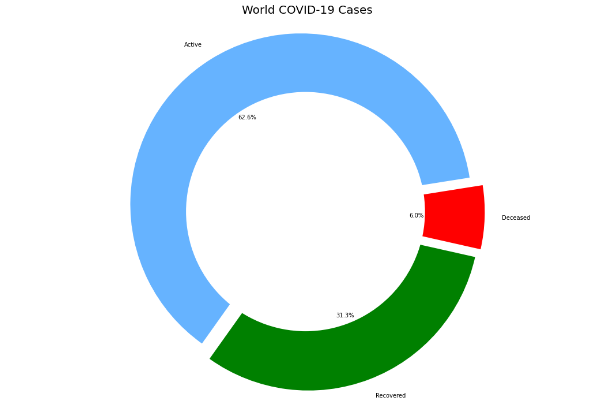
ICMR\_details = ICMR\_details.dropna(subset=['TotalSamplesTested', 'TotalPositiveCases'])

We will import our datasets to environment and name a unique variable for all the dataset we have uploaded, (some datasets are uploaded manually .csv files, some are downloaded directly from GitHub.)

**1. World Updates**

world\_confirmed = confirmed\_df[confirmed\_df.columns[-1:]].sum() world\_recovered = recovered\_df[recovered\_df.columns[-1:]].sum() world\_deaths = deaths\_df[deaths\_df.columns[-1:]].sum() world\_active = world\_confirmed - (world\_recovered - world\_deaths) labels = ['Active','Recovered','Deceased'] sizes = [world\_active,world\_recovered,world\_deaths] color= ['#66b3ff','green','red'] explode = [] **for** i **in** labels: explode.append(0.05) plt.figure(figsize= (15,10)) plt.pie(sizes, labels=labels, autopct='**%1.1f%%**', startangle=9, explode =explode,colors = color) centre\_circle = plt.Circle((0,0),0.70,fc='white') fig = plt.gcf() fig.gca().add\_artist(centre\_circle) plt.title('World COVID-19 Cases',fontsize = 20) plt.axis('equal') plt.tight\_layout()

the above code are preprocessing codes to plot a graph on world current updates covid19. (line by line code is explained in the Presentation Video)



The Above output is the result of world current updates preprocessing code, The results says that there are currently 62.6% of active corona patients, 31.3% of corona patients are recovered from confirmed patients, and 6 % of confirmed patients have deceased.

hotspots = ['India','US','United Kingdom']

dates = list(confirmed\_df.columns[4:])

dates = list(pd.to\_datetime(dates))

dates\_india = dates[8:]

df1 = confirmed\_df.groupby('Country/Region').sum().reset\_index()

df2 = deaths\_df.groupby('Country/Region').sum().reset\_index()

df3 = recovered\_df.groupby('Country/Region').sum().reset\_index()

global\_confirmed = {}

global\_deaths = {}

global\_recovered = {}

global\_active= {}

**for** country **in** hotspots:

k =df1[df1['Country/Region'] == country].loc[:,'1/30/20':]

global\_confirmed[country] = k.values.tolist()[0]

k =df2[df2['Country/Region'] == country].loc[:,'1/30/20':]

global\_deaths[country] = k.values.tolist()[0]

k =df3[df3['Country/Region'] == country].loc[:,'1/30/20':]

global\_recovered[country] = k.values.tolist()[0]

**for** country **in** hotspots:

k = list(map(int.\_\_sub\_\_, global\_confirmed[country], global\_deaths[country]))

global\_active[country] = list(map(int.\_\_sub\_\_, k, global\_recovered[country]))

fig = plt.figure(figsize= (15,15))

plt.suptitle('Active, Recovered, Deaths in Hotspot Countries and India as of May 12',fontsize = 20,y=1.0)

*#plt.legend()*

k=0

**for** i **in** range(1,4):

ax = fig.add\_subplot(6,2,i)

ax.xaxis.set\_major\_formatter(mdates.DateFormatter('**%d**-%b'))

ax.bar(dates\_india,global\_active[hotspots[k]],color = 'green',alpha = 0.6,label = 'Active');

ax.bar(dates\_india,global\_recovered[hotspots[k]],color='grey',label = 'Recovered');

ax.bar(dates\_india,global\_deaths[hotspots[k]],color='red',label = 'Death');

plt.title(hotspots[k])

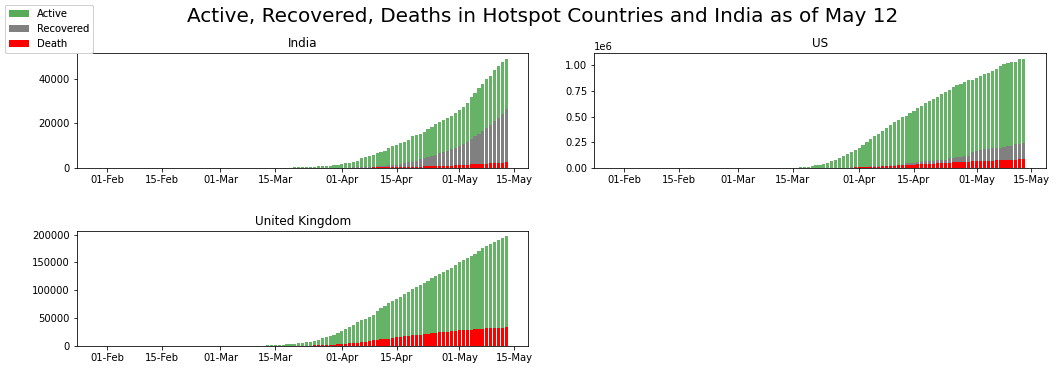
handles, labels = ax.get\_legend\_handles\_labels()

fig.legend(handles, labels, loc='upper left')

k=k+1

plt.tight\_layout(pad=3.0)

the Above line of codes will help to plot the Corona updates for US, UK and INDIA (latest 12th may 2020)



the above plot is self-explanatory, US has the greater number of confirmed corona patients as it has reached above 1 million, In United Kingdom Surprisingly NO data available for recovered corona patients, for more details we can go through this article ,India have reached close to 60,000 as of 12th may.

# 2 . India Updates

labels = list(age\_details['AgeGroup'])

sizes = list(age\_details['TotalCases'])

explode = []

**for** i **in** labels:

explode.append(0.05)

plt.figure(figsize= (15,10))

plt.pie(sizes, labels=labels, autopct='**%1.1f%%**', startangle=9, explode =explode)

centre\_circle = plt.Circle((0,0),0.70,fc='white')

fig = plt.gcf()

fig.gca().add\_artist(centre\_circle)

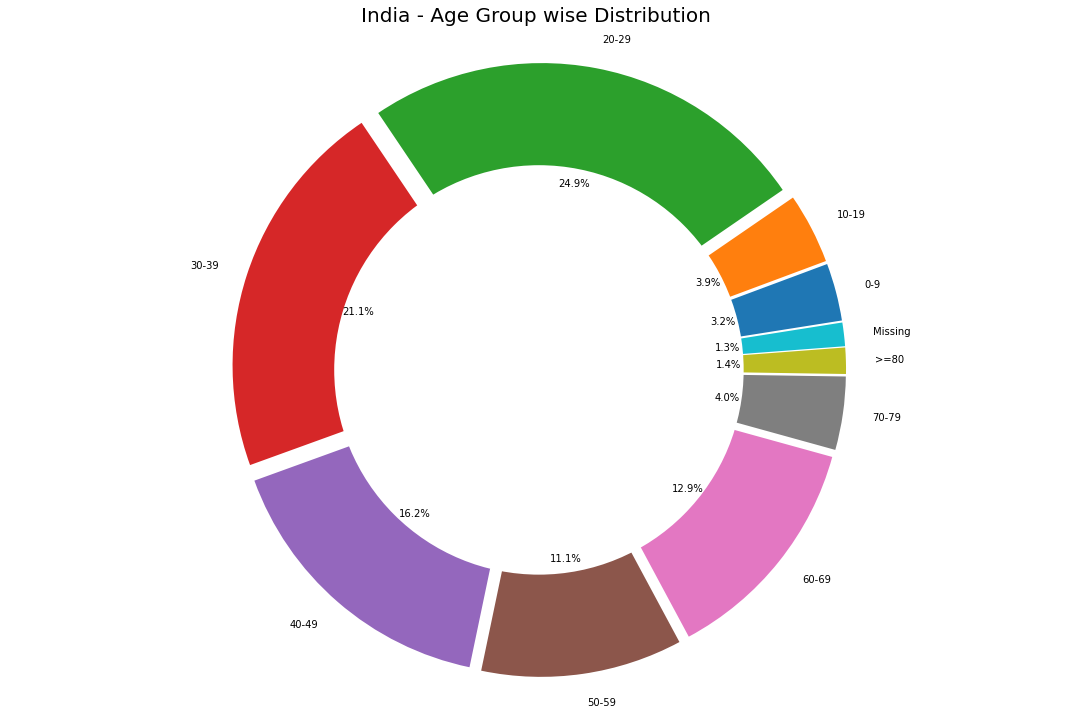
plt.title('India - Age Group wise Distribution',fontsize = 20)

plt.axis('equal')

plt.tight\_layout()

### The above line of codes will help to plot the Age Group wise Distribution of India,

The output of the codes is.

 We could see that the **age group <40 is the most affected** which is against the trend which says elderly people are more at risk of being affected. Only 17% of people >60 are affected.

labels = ['Missing', 'Male', 'Female']

sizes = []

sizes.append(individual\_details['gender'].isnull().sum())

sizes.append(list(individual\_details['gender'].value\_counts())[0])

sizes.append(list(individual\_details['gender'].value\_counts())[1])

explode = (0, 0.1, 0)

colors = ['#ffcc99','#66b3ff','#ff9999']

plt.figure(figsize= (15,10))

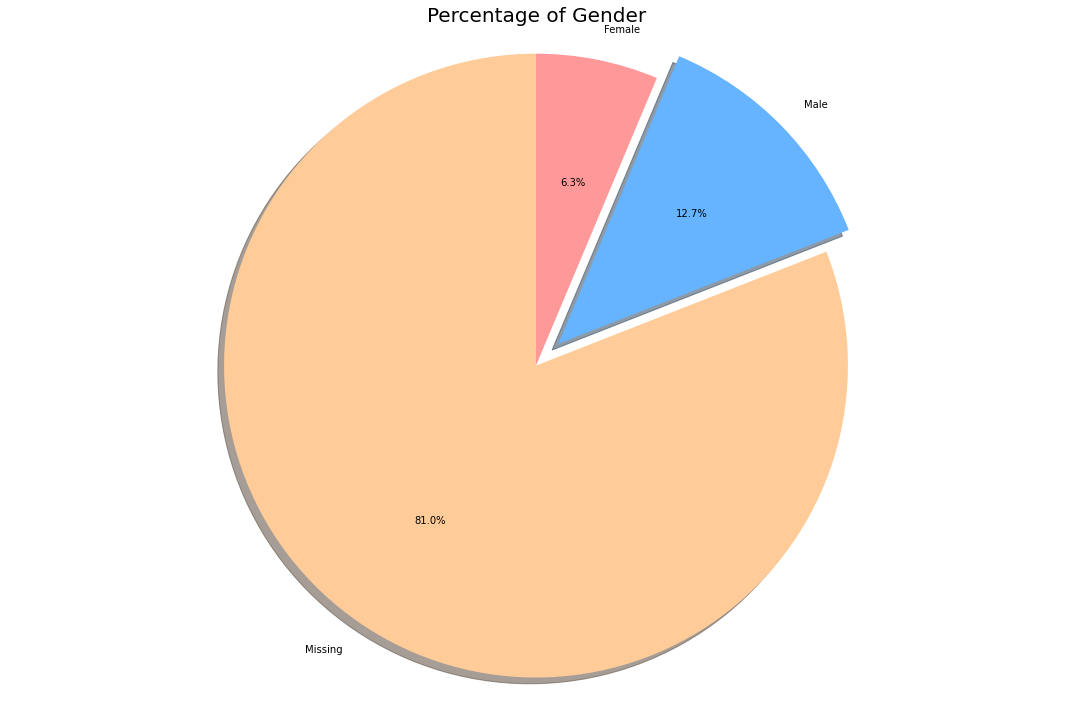
plt.title('Percentage of Gender',fontsize = 20)

plt.pie(sizes, explode=explode, labels=labels, colors=colors, autopct='**%1.1f%%**',shadow=**True**, startangle=90)

plt.axis('equal')

plt.tight\_layout()

the above will help to create pie chart using ['Missing', 'Male', 'Female'] as labels,

 **80% of the patient’s gender information is missing. Let's analyses with remaining the data.**

labels = ['Male', 'Female']

sizes = []

sizes.append(list(individual\_details['gender'].value\_counts())[0])

sizes.append(list(individual\_details['gender'].value\_counts())[1])

explode = (0.1, 0)

colors = ['#66b3ff','#ff9999']

plt.figure(figsize= (15,10))

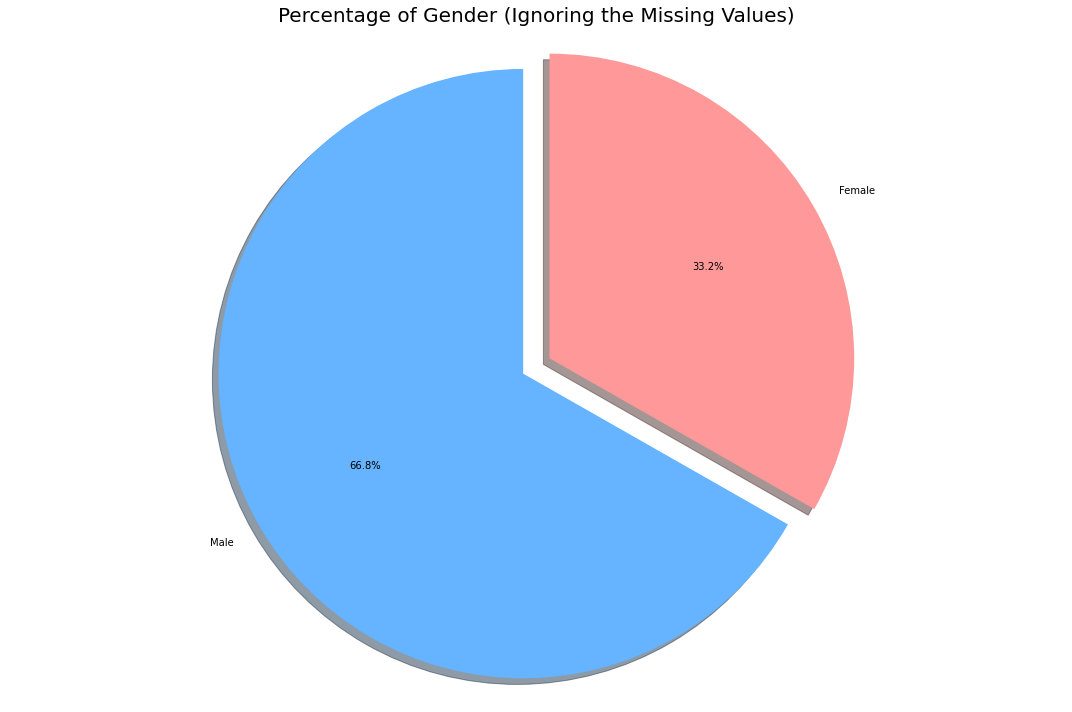
plt.pie(sizes, explode=explode, labels=labels, colors=colors, autopct='**%1.1f%%**',

shadow=**True**, startangle=90)

plt.title('Percentage of Gender (Ignoring the Missing Values)',fontsize = 20)

plt.axis('equal')

plt.tight\_layout()

   
Out of 100% of available covid19 data, 81% of gender column data is missing, hence we will plot with available data

Men are the most affected accounting to 67%, female 33% , But remember we have ~80% data missing.

### **The Spike in India**

countries = ['China','US', 'Italy', 'Spain', 'France','India']

global\_confirmed = []

global\_recovered = []

global\_deaths = []

global\_active = []

**for** country **in** countries:

k =df1[df1['Country/Region'] == country].loc[:,'1/30/20':]

global\_confirmed.append(k.values.tolist()[0])

k =df2[df2['Country/Region'] == country].loc[:,'1/30/20':]

global\_deaths.append(k.values.tolist()[0])

k =df3[df3['Country/Region'] == country].loc[:,'1/30/20':]

global\_deaths.append(k.values.tolist()[0])

plt.figure(figsize= (15,10))

plt.xticks(rotation = 90 ,fontsize = 11)

plt.yticks(fontsize = 10)

plt.xlabel("Dates",fontsize = 20)

plt.ylabel('Total cases',fontsize = 20)

plt.title("Comparison with other Countries" , fontsize = 20)

**for** i **in** range(len(countries)):

plt.plot\_date(y= global\_confirmed[i],x= dates\_india,label = countries[i],linestyle ='-')

plt.legend();

### the above code will help to plot a comparison of confirmed cases between 'China','US', 'Italy', 'Spain', 'France','India'

### C:\Users\sanjay\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\E476171A.tmp

### US is at peak, other countries are normal,

### Though being highly populated the relative confirmed cases of India is low compared to other countries. This could be because of two reasons

1. 21 day lockdown imposed by prime minister Narendra Modi (Source : [Health Ministry](http://www.indiatoday.in/india/story/without-lockdown-india-would-have-2-lakh-covid-19-cases-by-now-health-ministry-1665903-2020-04-11))
2. Low testing rate (Source: [news18](https://www.news18.com/news/india/with-just-18-tests-per-million-of-population-why-indias-low-testing-rate-for-coronavirus-is-worrying-2552845.html))

ICMR\_details['Percent\_positive'] = round((ICMR\_details['TotalPositiveCases']/ICMR\_details['TotalSamplesTested'])\*100,1)

fig, ax1 = plt.subplots(figsize= (15,5))

ax1.xaxis.set\_major\_formatter(mdates.DateFormatter('**%d**-%b'))

ax1.set\_ylabel('Positive Cases (**% o**f Total Samples Tested)')

ax1.bar(ICMR\_details['DateTime'] , ICMR\_details['Percent\_positive'], color="red",label = 'Percentage of Positive Cases')

ax1.text(ICMR\_details['DateTime'][0],4, 'Total Samples Tested as of Apr 23rd = 541789', style='italic',fontsize= 10,

bbox={'facecolor': 'white' ,'alpha': 0.5, 'pad': 5})

ax2 = ax1.twinx()

ax2.xaxis.set\_major\_formatter(mdates.DateFormatter('**%d**-%b'))

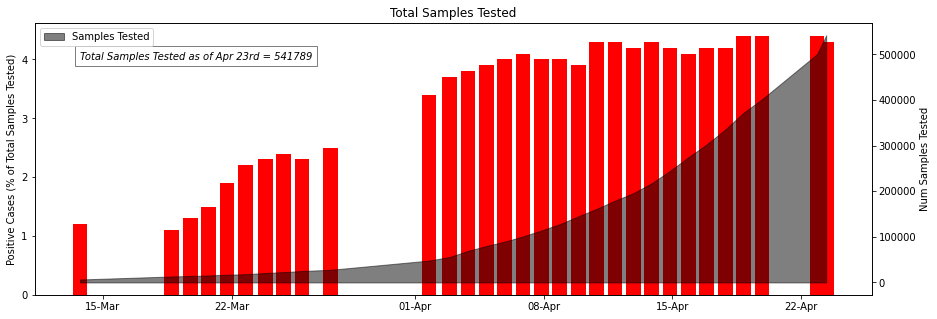
ax2.set\_ylabel('Num Samples Tested')

ax2.fill\_between(ICMR\_details['DateTime'],ICMR\_details['TotalSamplesTested'],color = 'black',alpha = 0.5,label = 'Samples Tested');

plt.legend(loc="upper left")

plt.title('Total Samples Tested')

plt.show()



The Indian Council of Medical Research (ICMR) recorded 21,797 coronavirus (Covid-19) cases in India by Thursday. A total of 500,542 samples from 485,172 individuals have been tested as on April 23, testing samples is rapidly growing day by day,

## **Statewise Insights**

state\_cases = india\_covid\_19.groupby('State/UnionTerritory')['Confirmed','Deaths','Cured'].max().reset\_index()

*#state\_cases = state\_cases.astype({'Deaths': 'int'})*

state\_cases['Active'] = state\_cases['Confirmed'] - (state\_cases['Deaths']+state\_cases['Cured'])

state\_cases["Death Rate (per 100)"] = np.round(100\*state\_cases["Deaths"]/state\_cases["Confirmed"],2)

state\_cases["Cure Rate (per 100)"] = np.round(100\*state\_cases["Cured"]/state\_cases["Confirmed"],2)

state\_cases.sort\_values('Confirmed', ascending= **False**).fillna(0).style.background\_gradient(cmap='Blues',subset=["Confirmed"])\

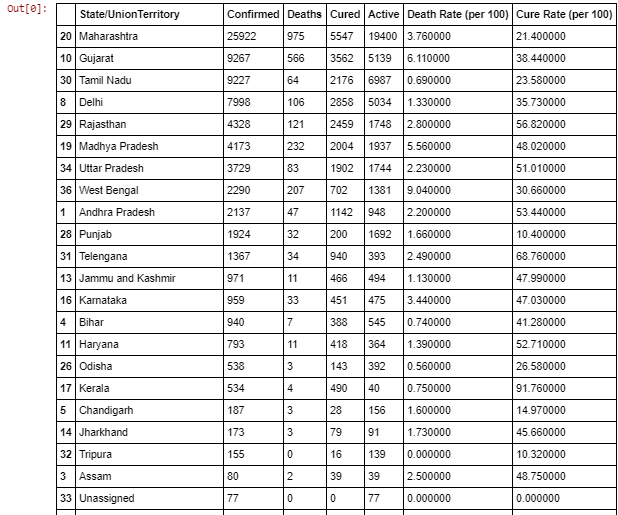
.background\_gradient(cmap='Blues',subset=["Deaths"])\

.background\_gradient(cmap='Blues',subset=["Cured"])\

.background\_gradient(cmap='Blues',subset=["Active"])\

.background\_gradient(cmap='Blues',subset=["Death Rate (per 100)"])\

.background\_gradient(cmap='Blues',subset=["Cure Rate (per 100)"])



There are many states like Maharashtra, Delhi, Madhya Pradesh, Rajasthan, Gujrat, Uttar Pradesh, and West Bengal, who are still at high risk. These states may see a huge jump in confirmed COVID-19 cases in the coming days if preventive measures are not implemented properly. On the positive side, Kerala has shown how to effectively “flatten” or even “crush the curve” of COVID-19 cases. We hope India can be free of COVID19 with a strong determination as already shown by the central and respective state Governments.

colors\_list = ['cyan','teal']

states = individual\_details['detected\_state'].unique()

**if** len(states)%2==0:

n\_rows = int(len(states)/2)

**else**:

n\_rows = int((len(states)+1)/2)

plt.figure(figsize=(14,60))

**for** idx,state **in** enumerate(states):

plt.subplot(n\_rows,2,idx+1)

y\_order = individual\_details[individual\_details['detected\_state']==state]['detected\_district'].value\_counts().index

**try**:

g = sns.countplot(data=individual\_details[individual\_details['detected\_state']==state],y='detected\_district',orient='v',color=colors\_list[idx%2],order=y\_order)

plt.xlabel('Number of Cases')

plt.ylabel('')

plt.title(state)

plt.ylim(14,-1)

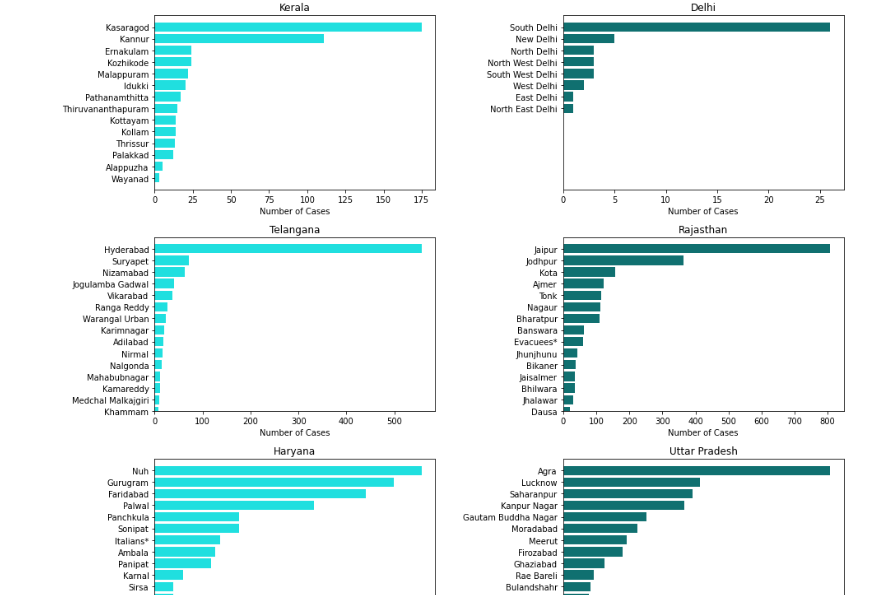
**except**:

**pass**

plt.tight\_layout()

plt.show()

the above will try to plot all the state



1. Maharashtra- The western state, among the country’s most developed, has the largest number of coronavirus patients at 23,264 positive cases including 3,470 discharged patients and 731 deaths.

2. Gujarat – Maharashtra’s neighboring state and also among the most industrialized, it has registered 9,723 total positive cases, out of which 1,872 have been cured, while 449 died.

3. Delhi – The country’s capital is third on the list with 8,406 positive cases so far, including 2020 recovered patients and 68 fatalities.

4. Tamil Nadu- The worst affected southern state has over 7,654 positive cases including 1,605 recoveries and 40 casualties.

5. Rajasthan- The western state has 5,596 cases, including 1,916 cured patients and 101 deaths.

6. Madhya Pradesh- The state is sixth on the list with 5,421 positive cases including 1,349 recoveries and 200 deaths.

7. Uttar Pradesh- India’s most populous state has just under 5,000 cases at 4,667, including 1,387 cured patients and 66 casualties.

8. Andhra Pradesh- With 2,770 positive cases, Andhra is the second-worst affected state in southern India. 842 patients have been discharged in the state while 41 have died so far.

9. West Bengal- It is the worst affected state in the east with 2,202 positive cases including 364 people who were cured and 160 who died

10. Punjab- The northern state has 1,912 cases including 152 recoveries and 29 deaths.

### Statewise Testing and Healthcare Insights

hospital\_beds =hospital\_beds.drop([36])

cols\_object = list(hospital\_beds.columns[2:8])

**for** cols **in** cols\_object:

hospital\_beds[cols] = hospital\_beds[cols].astype(int,errors = 'ignore')

top\_10\_primary = hospital\_beds.nlargest(10,'NumPrimaryHealthCenters\_HMIS')

top\_10\_community = hospital\_beds.nlargest(10,'NumCommunityHealthCenters\_HMIS')

top\_10\_district\_hospitals = hospital\_beds.nlargest(10,'NumDistrictHospitals\_HMIS')

top\_10\_public\_facility = hospital\_beds.nlargest(10,'TotalPublicHealthFacilities\_HMIS')

top\_10\_public\_beds = hospital\_beds.nlargest(10,'NumPublicBeds\_HMIS')

plt.figure(figsize=(15,10))

plt.suptitle('Top 10 States in each Health Facility',fontsize=20)

plt.subplot(221)

plt.title('Primary Health Centers')

plt.barh(top\_10\_primary['State/UT'],top\_10\_primary['NumPrimaryHealthCenters\_HMIS'],color ='#87479d');

plt.subplot(222)

plt.title('Community Health Centers')

plt.barh(top\_10\_community['State/UT'],top\_10\_community['NumCommunityHealthCenters\_HMIS'],color = '#9370db');

plt.subplot(224)

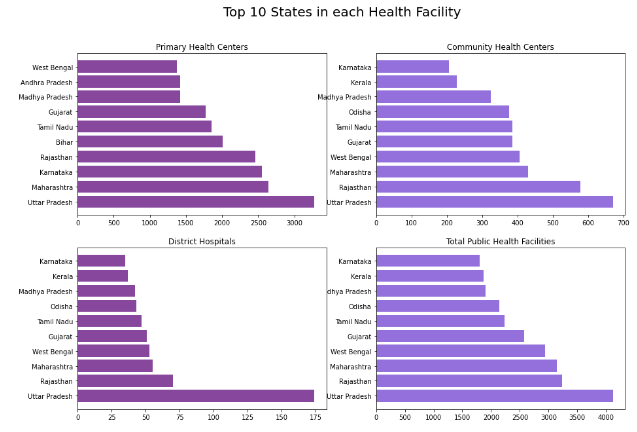
plt.title('Total Public Health Facilities')

plt.barh(top\_10\_community['State/UT'],top\_10\_public\_facility['TotalPublicHealthFacilities\_HMIS'],color='#9370db');

plt.subplot(223)

plt.title('District Hospitals')

plt.barh(top\_10\_community['State/UT'],top\_10\_district\_hospitals['NumDistrictHospitals\_HMIS'],color = '#87479d');



Uttar Pradesh have highest Primary Health Centers, Community Health Centers, Total Public Health Facilities, District Hospitals

Rajasthan is 2nd highest in Community Health Centers, Total Public Health Facilities, District Hospitals

state\_test = pd.pivot\_table(state\_testing, values=['TotalSamples','Negative','Positive'], index='State', aggfunc='max')

state\_names = list(state\_test.index)

state\_test['State'] = state\_names

plt.figure(figsize=(15,10))

sns.set\_color\_codes("pastel")

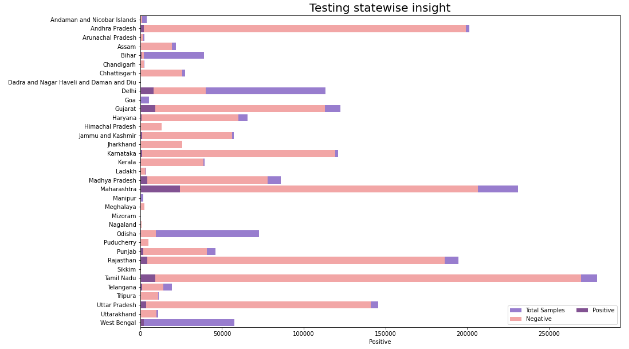
sns.barplot(x="TotalSamples", y= state\_names, data=state\_test,label="Total Samples", color = '#9370db')

sns.barplot(x='Negative', y=state\_names, data=state\_test,label='Negative', color= '#ff9999')

sns.barplot(x='Positive', y=state\_names, data=state\_test,label='Positive', color='#87479d')

plt.title('Testing statewise insight',fontsize = 20)

plt.legend(ncol=2, loc="lower right", frameon=**True**);



Above Bar Graph is self-Explanatory, Tamil Nadu have done highest testing samples, and have more negative corona results,

values = list(ICMR\_labs['state'].value\_counts())

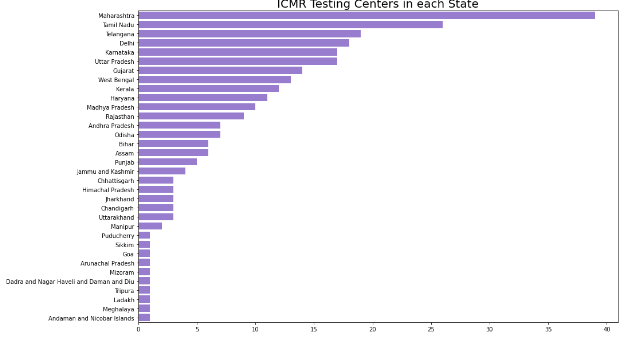
names = list(ICMR\_labs['state'].value\_counts().index)

plt.figure(figsize=(15,10))

sns.set\_color\_codes("pastel")

plt.title('ICMR Testing Centers in each State', fontsize = 20)

sns.barplot(x= values, y= names,color = '#9370db');



Maharashtra, Tamil Nadu have highest ICMR testing centers in India

# Prediction

## Prediction using Prophet Model

Prophet is a procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects. It works best with time series that have strong seasonal effects and several seasons of historical data. Prophet is robust to missing data and shifts in the trend, and typically handles outliers well.

k = df1[df1['Country/Region']=='India'].loc[:,'1/22/20':]

india\_confirmed = k.values.tolist()[0]

data = pd.DataFrame(columns = ['ds','y'])

data['ds'] = dates

data['y'] = india\_confirmed

prop=Prophet()

prop.fit(data)

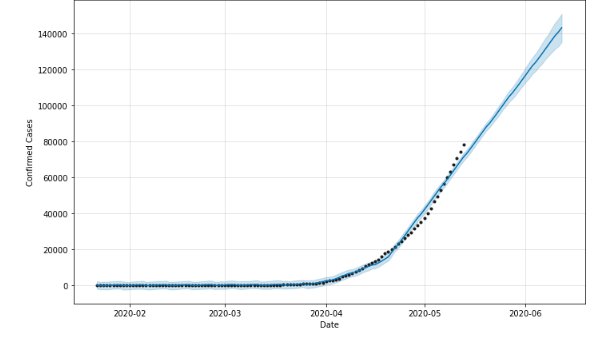
future=prop.make\_future\_dataframe(periods=30)

prop\_forecast=prop.predict(future)

forecast = prop\_forecast[['ds','yhat']].tail(30)

fig = plot\_plotly(prop, prop\_forecast)

fig = prop.plot(prop\_forecast,xlabel='Date',ylabel='Confirmed Cases')



For the month June/2020-06 we have predicted that there will be 1,20,000 in India using Prophet Model

## Prediction using ML Models = 'LGBM', 'Random Forest', 'XGBoost'

**LGBR** = Light GBM is a fast, distributed, high-performance gradient boosting framework based on decision tree algorithm, used for ranking, classification and many other machine learning tasks.

**Random Forest** = The random forest is a classification algorithm consisting of many decisions trees. It uses bagging and feature randomness when building each individual tree to try to create an uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree.

**XGBoost** stands for “Extreme Gradient Boosting”, where the term “Gradient Boosting” originates from the paper Greedy Function Approximation: A Gradient Boosting Machine, by Friedman. ... We think this explanation is cleaner, more formal, and motivates the model formulation used in XGBoost.

test = pd.read\_csv('test.csv')

train['Date'] = pd.to\_datetime(train['Date'])

test['Date'] = pd.to\_datetime(test['Date'])

train['day'] = train['Date'].dt.day

train['month'] = train['Date'].dt.month

train['dayofweek'] = train['Date'].dt.dayofweek

train['dayofyear'] = train['Date'].dt.dayofyear

train['quarter'] = train['Date'].dt.quarter

train['weekofyear'] = train['Date'].dt.weekofyear

test['day'] = test['Date'].dt.day

test['month'] = test['Date'].dt.month

test['dayofweek'] = test['Date'].dt.dayofweek

test['dayofyear'] = test['Date'].dt.dayofyear

test['quarter'] = test['Date'].dt.quarter

test['weekofyear'] = test['Date'].dt.weekofyear

countries = list(train['Country\_Region'].unique())

india\_code = countries.index('India')

train = train.drop(['Date','Id'],1)

test = test.drop(['Date'],1)

train.Province\_State.fillna('NaN', inplace=**True**)

oe = OrdinalEncoder()

train[['Province\_State','Country\_Region']] = oe.fit\_transform(train.loc[:,['Province\_State','Country\_Region']])

test.Province\_State.fillna('NaN', inplace=**True**)

oe = OrdinalEncoder()

test[['Province\_State','Country\_Region']] = oe.fit\_transform(test.loc[:,['Province\_State','Country\_Region']])

columns = ['day','month','dayofweek','dayofyear','quarter','weekofyear','Province\_State', 'Country\_Region','ConfirmedCases','Fatalities']

test\_columns = ['day','month','dayofweek','dayofyear','quarter','weekofyear','Province\_State','Country\_Region']

train = train[columns]

x = train.drop(['Fatalities','ConfirmedCases'], 1)

y = train['ConfirmedCases']

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2,random\_state=0)

test = test[test\_columns]

test\_india = test[test['Country\_Region'] == india\_code]

models = []

mse = []

mae = []

rmse = []

we will import the data, preprocess it and spit it for training and testing with split size 0.2.

the columns for x/independent variables will be 'day','month','dayofweek','dayofyear','quarter','weekofyear','Province\_State','Country\_Region'

the dependent / target / y will be 'ConfirmedCases'

*#change date*

day = 14

month = 5

dayofweek = 2

dayofyear = 134

quarter = 2

weekofyear = 20

*#72 Province\_State and 78 Country\_Region are the values for india , dont change*

Province\_State = 72.0

Country\_Region = 78.0

In the above codes we can change the values of day, month etc ... As of now we will try to predict for 14/5

lgbm = LGBMRegressor(n\_estimators=1300)

lgbm.fit(x\_train,y\_train)

pred = lgbm.predict(x\_test)

models.append('LGBM')

mse.append(round(mean\_squared\_error(pred, y\_test),2))

mae.append(round(mean\_absolute\_error(pred, y\_test),2))

rmse.append(round(np.sqrt(mean\_squared\_error(pred, y\_test)),2))

lgbm\_pred = lgbm.predict([[ day,month ,dayofweek ,dayofyear ,quarter ,weekofyear , Province\_State,Country\_Region ]])[0] print ("On Date = ", day,'/',month,'/','2020', "**\n**",'Total Confirmed Cases will be =',lgbm\_pred ,"**\n**","(Using LGBM model)" )

On Date = 14 / 5 / 2020

Total Confirmed Cases will be = 62426.92894163738

(Using LGBM model)

## Random Forest Regressor

rf = RandomForestRegressor(n\_estimators=100)

rf.fit(x\_train,y\_train)

pred = rf.predict(x\_test)

rfr\_forecast = rf.predict(test\_india)

models.append('Random Forest')

mse.append(round(mean\_squared\_error(pred, y\_test),2))

mae.append(round(mean\_absolute\_error(pred, y\_test),2))

rmse.append(round(np.sqrt(mean\_squared\_error(pred, y\_test)),2))

rf\_pred = rf.predict([[day,month,dayofweek,dayofyear,quarter,weekofyear,Province\_State,Country\_Region]])[0] print ("On Date = ", day,'/',month,'/','2020', "**\n**",'Total Confirmed Cases will be =',rf\_pred ,"**\n**","(Using Random Forest Model)" )

On Date = 14 / 5 / 2020

Total Confirmed Cases will be = 61005.71

(Using Random Forest Model)

## XGB Regressor

xgb = XGBRegressor(n\_estimators=100)

xgb.fit(x\_train,y\_train)

pred = xgb.predict(x\_test)

xgb\_forecast = xgb.predict(test\_india)

models.append('XGBoost')

mse.append(round(mean\_squared\_error(pred, y\_test),2))

mae.append(round(mean\_absolute\_error(pred, y\_test),2))

rmse.append(round(np.sqrt(mean\_squared\_error(pred, y\_test)),2))

datacolumns = [[day, month,dayofweek,dayofyear,quarter,weekofyear,Province\_State,Country\_Region]]

data = pd.DataFrame(datacolumns, columns = ['day', 'month', 'dayofweek','dayofyear','quarter','weekofyear','Province\_State','Country\_Region'])

xgb\_pred = xgb.predict(data)

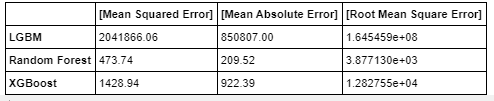
print ("On Date = ", day,'/',month,'/','2020', "**\n**",'Total Confirmed Cases will be =',xgb\_pred ,"**\n**","(Using XGB Regressor)" )

On Date = 14 / 5 / 2020

Total Confirmed Cases will be = [21000.07]

(Using XGB Regressor)

pd.DataFrame(index = models ,data=[mse,mae,rmse]).rename(columns={0:'[Mean Squared Error]',1:'[Mean Absolute Error]',2:'[Root Mean Square Error]'})



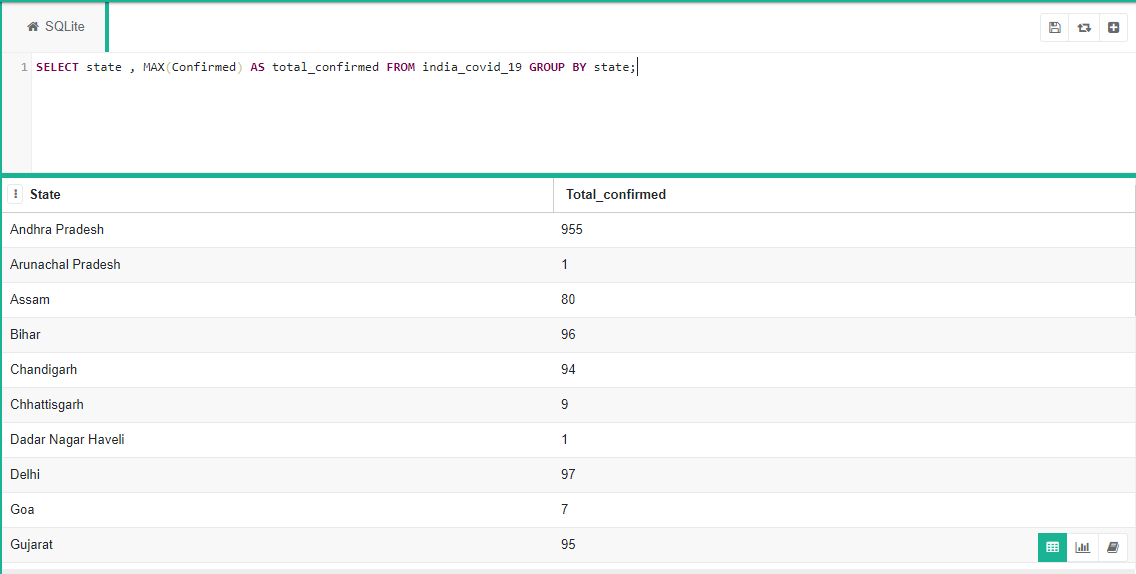
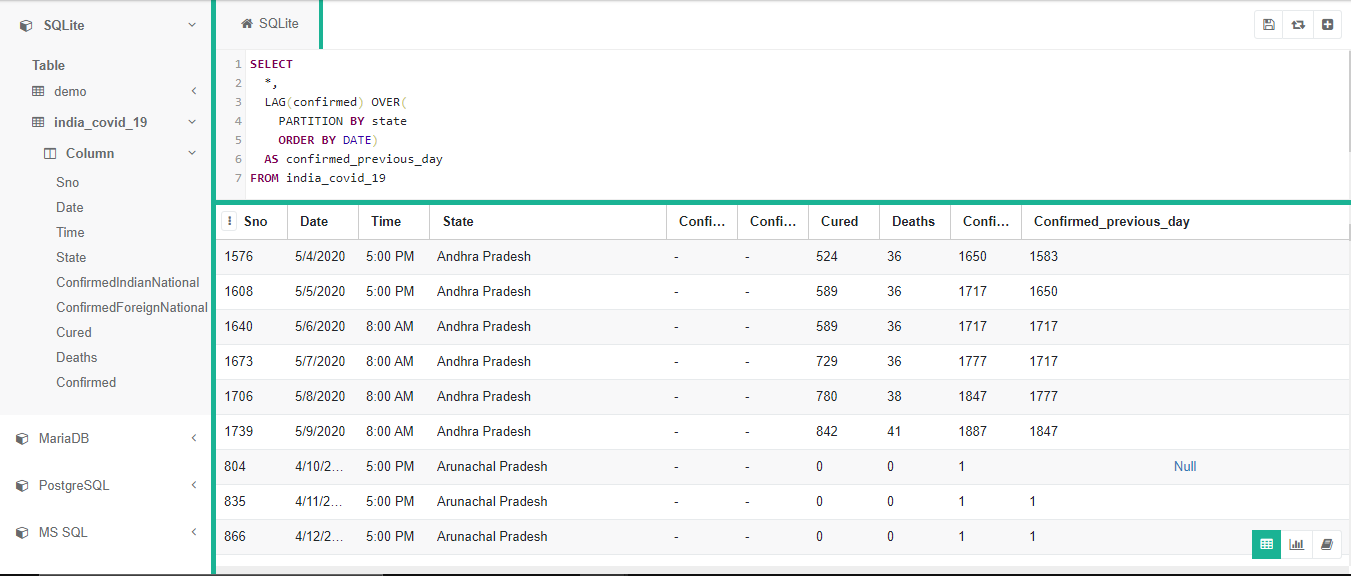
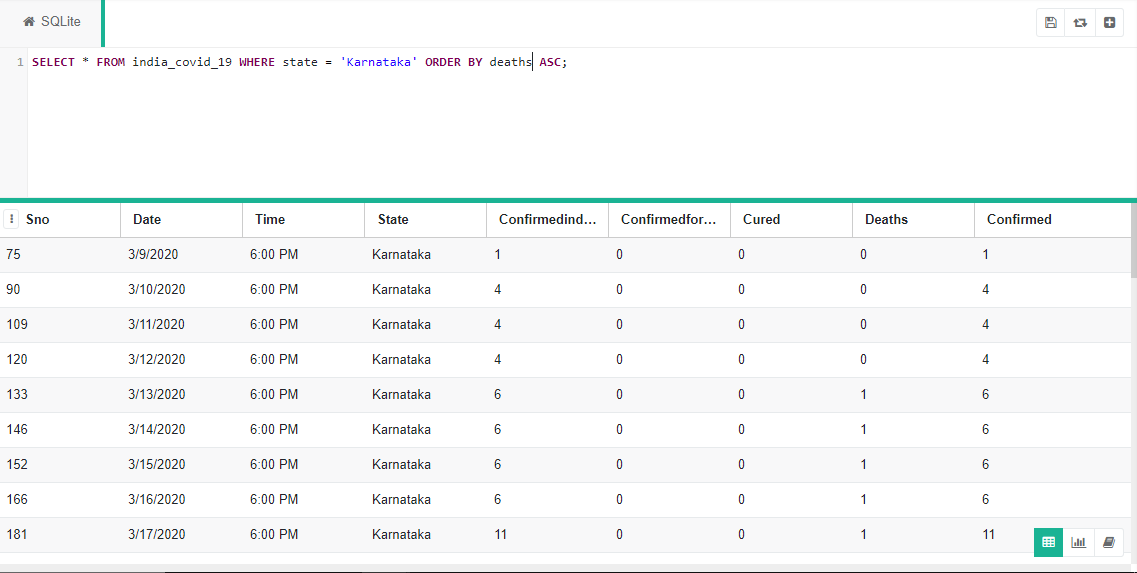
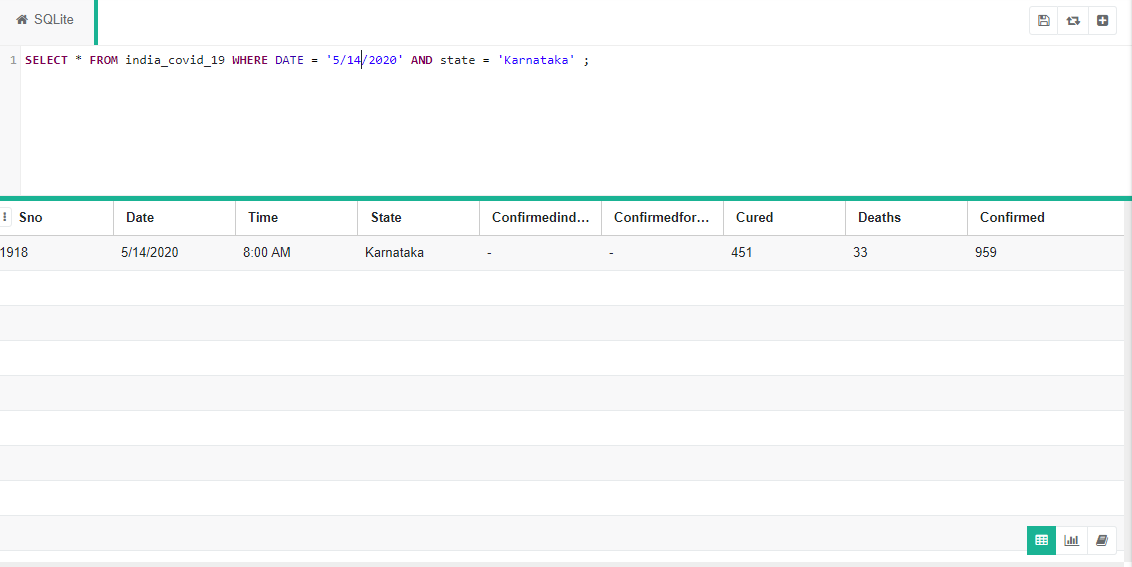
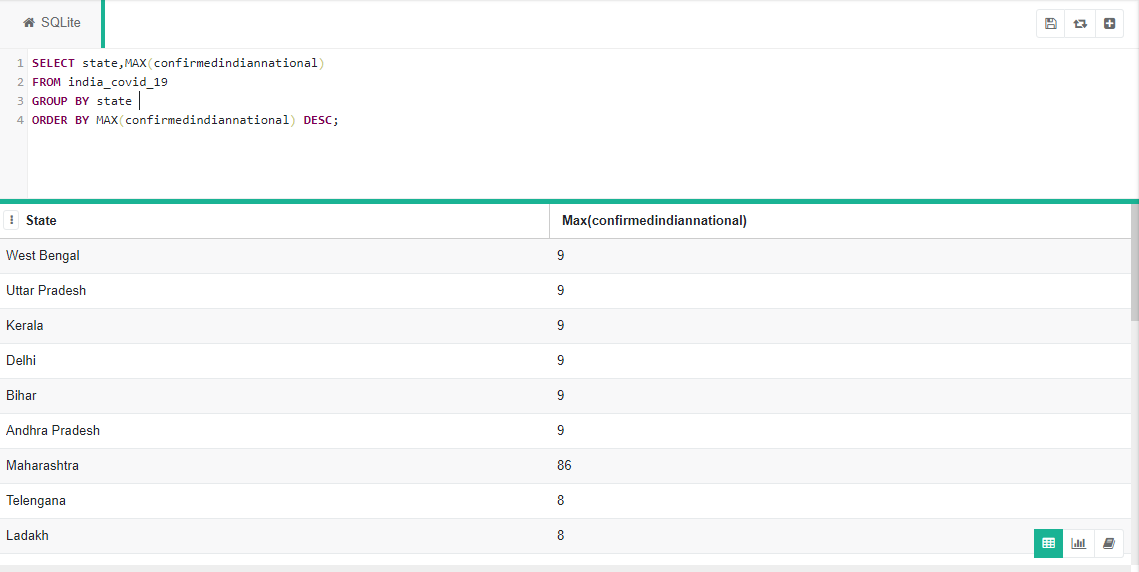
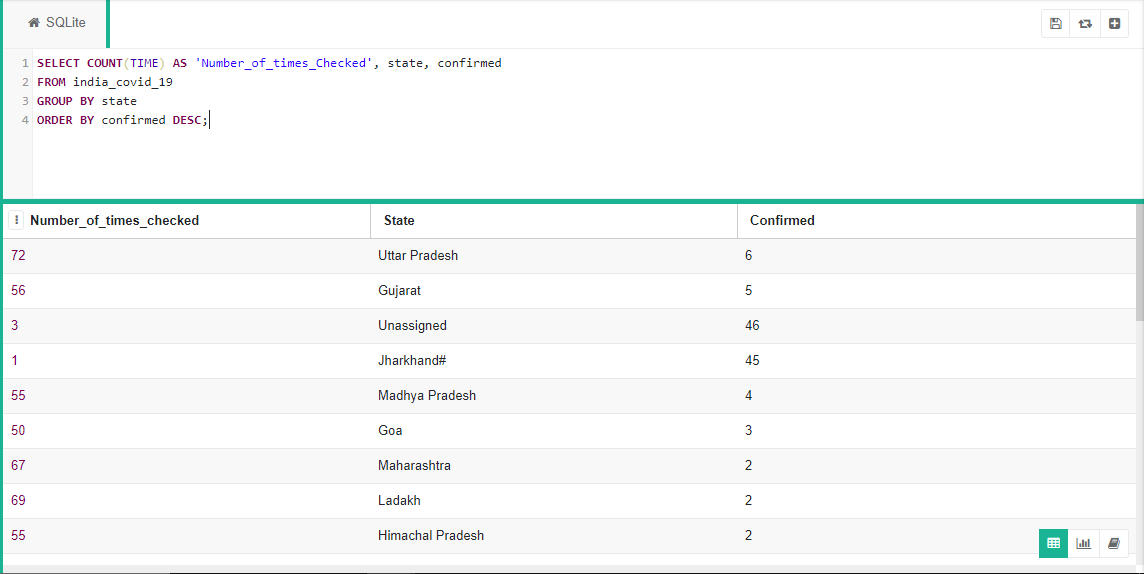
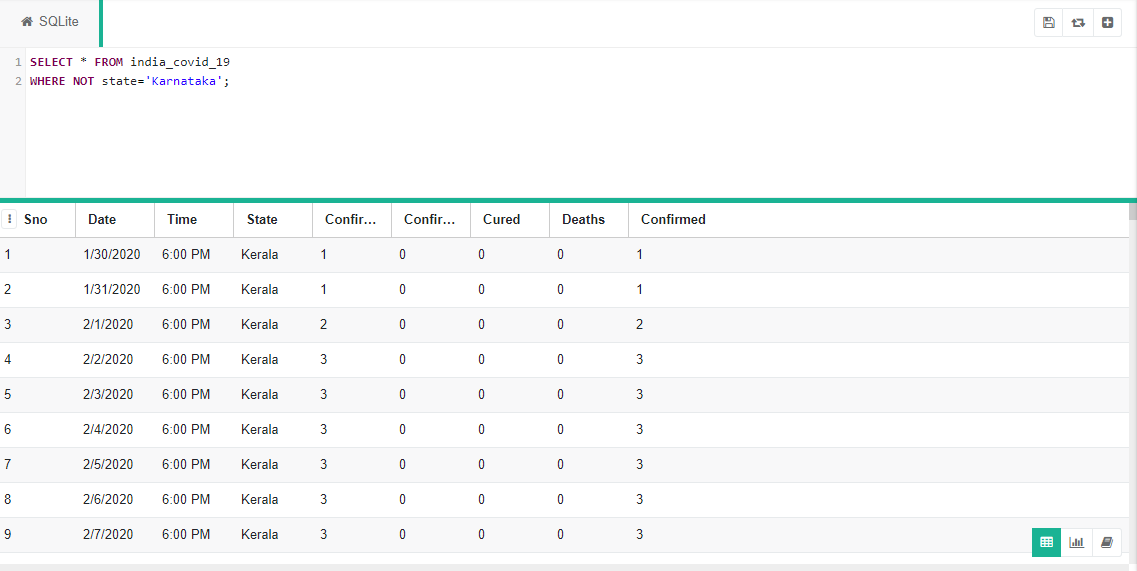
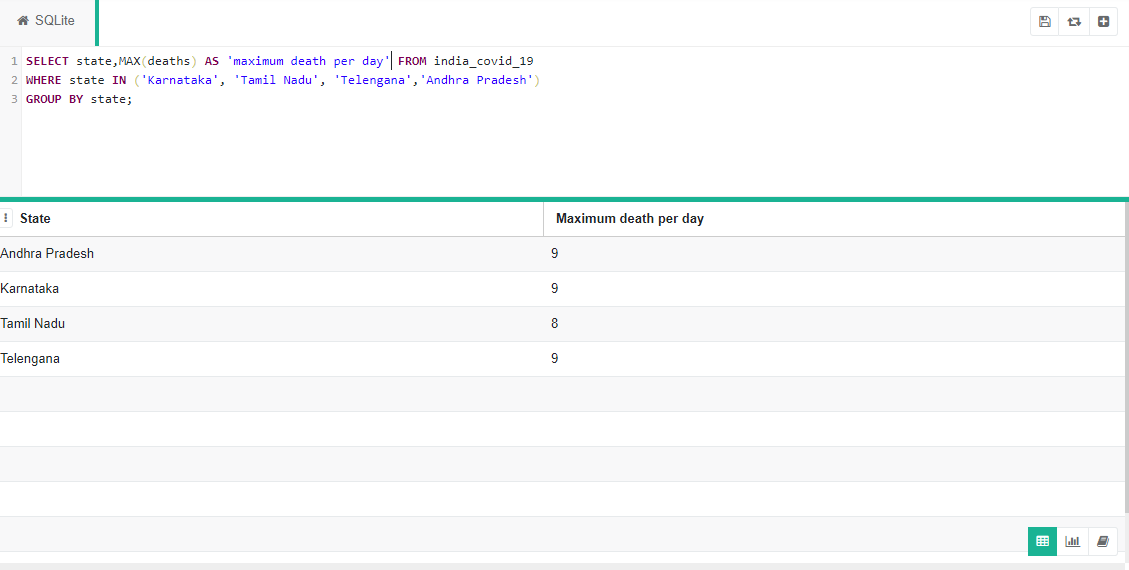
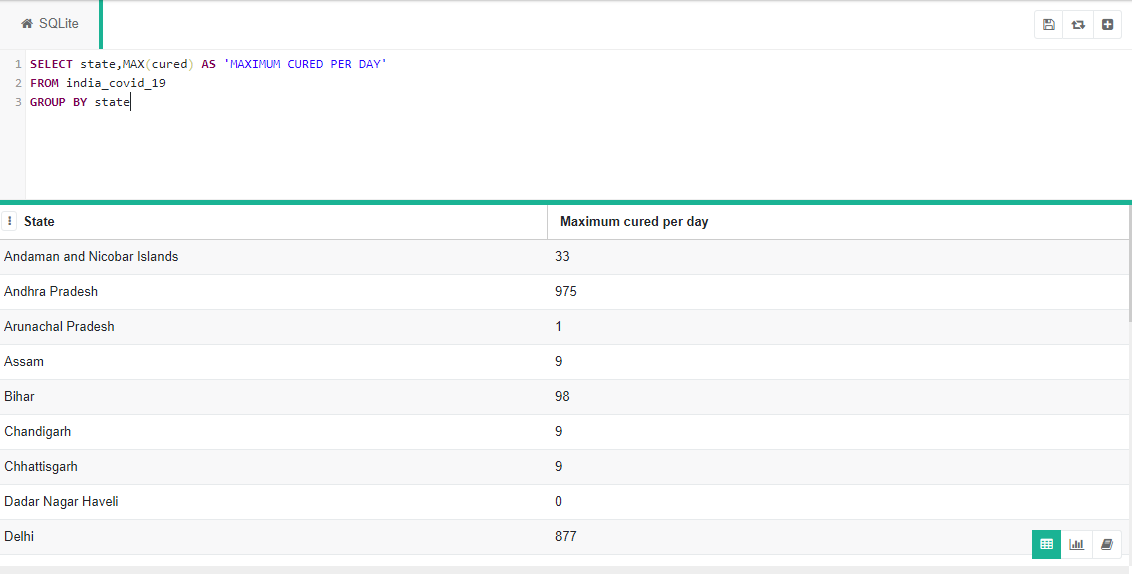
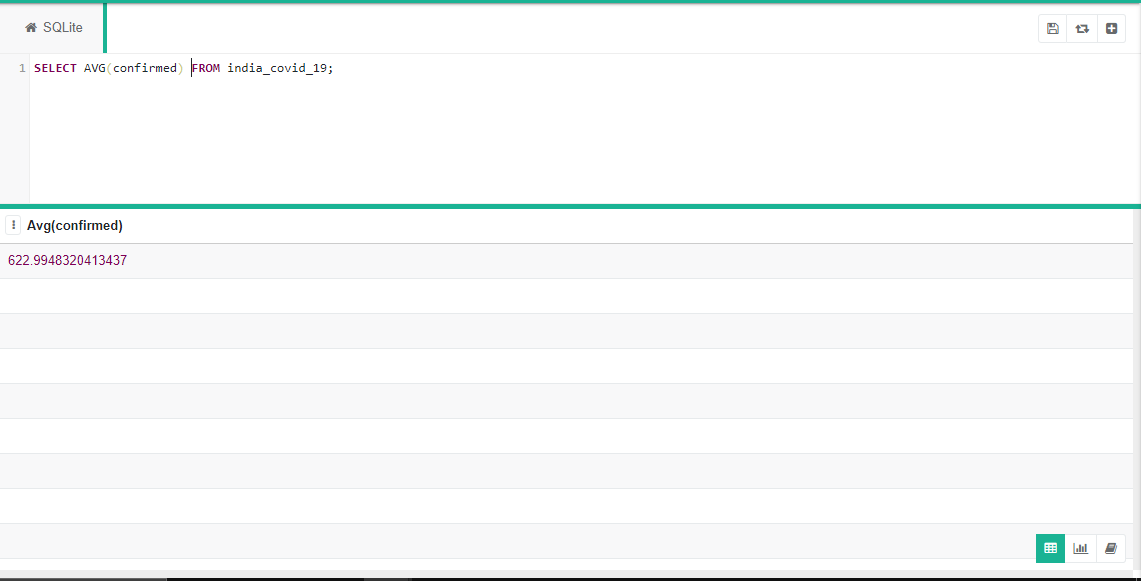
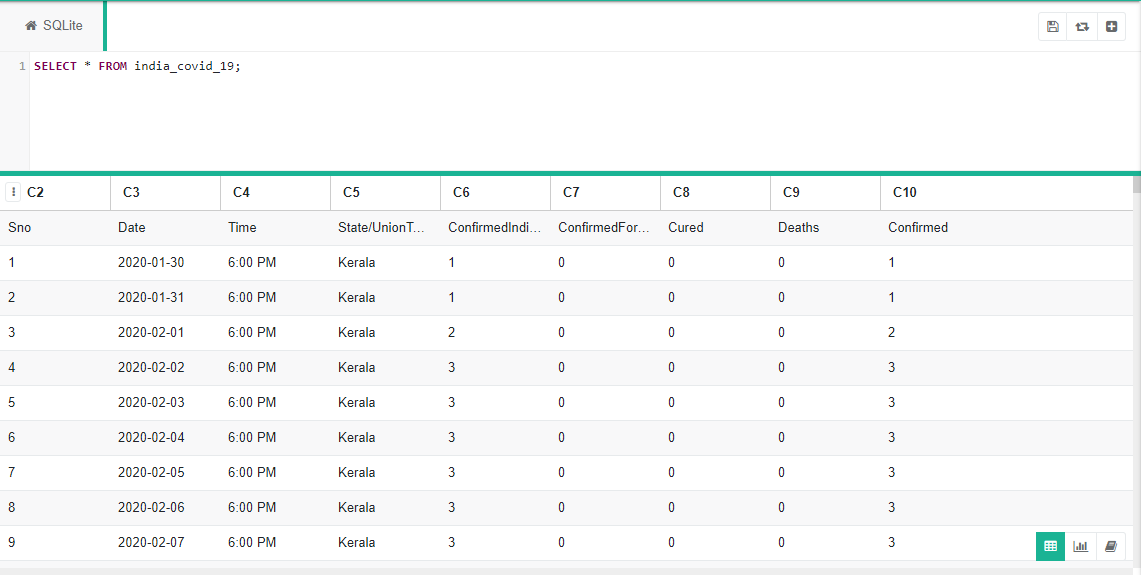
LGBM model is showing worst performance MSE, MAE is very high,

Random forest model is doing well as MSE and MAE is very less compared to other models, but the RMSE for this model is very high, hence there is a problem of over fitting and may be model is not generalized.

XGboost MSE and MAE is showing medium error, and RMSE is also less compared to 3 models

**SQL QUERIES**

11 queries screenshots



**References – source of the dataset downloaded from internet**

<https://data.humdata.org/dataset/novel-coronavirus-2019-ncov-cases>

<https://github.com/datasets/covid-19>

<https://github.com/datameet/covid19>

<https://www.kaggle.com/allen-institute-for-ai/CORD-19-research-challenge>