Programming for Analytics

Mini Project

DESCRIPTIVE STATISTICS, ML ALGORITHM, SQL QUERIES

FOR COVID19 DATASET

Prepared by

Mohith B

20182MBA0420

Presidency University

Submitted to

Dr K Balanagarajan

Associate Professor

GAIN MORE KNOWLEDGE
REACH GREATER HEIGHTS

SCHOOL OF MANAGEMENT
PRESIDENCY UNIVERSITY,
BENGALURU – 560 064

Table of Contents

Name: Mohith B ID.NO: 20182MBA0420

Python codes: https://github.com/mohithxoxo/Project/blob/master/PA_covid19_v2.ipynb

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.)

```
print("age_details","\n",age_details.describe(),"\n\n",
"india_covid_19","\n",india_covid_19.describe(),"\n\n",
"hospital_beds","\n",hospital_beds.describe(),"\n\n",
"individual_details","\n",individual_details.describe(),"\n\n",
"ICMR_details","\n",ICMR_details.describe(),"\n\n",
"ICMR_labs","\n",ICMR_labs.describe(),"\n\n",
"state_testing","\n",state_testing.describe(),"\n\n",
"population","\n",population.describe(),"\n\n",
"world_population","\n",world_population.describe(),"\n\n",
"confirmed_df","\n",confirmed_df.describe(),"\n\n",
"deaths_df","\n",deaths_df.describe(),"\n\n",
"recovered_df","\n",recovered_df.describe(),"\n\n",
"latest_data","\n",latest_data.describe())
```

Above codes will help to see descriptive statistics all the datasets we have imported, some outputs are below.

```
 india_covid_19
        Unnamed: 0          Sno        Cured       Deaths     Confirmed
count  1935.000000  1935.000000  1935.000000  1935.000000   1935.000000
mean    967.000000   968.000000   159.310078    20.173643    622.994832
std     558.730704   558.730704   468.278202    77.425935   1942.369977
min       0.000000     1.000000     0.000000     0.000000      0.000000
25%     483.500000   484.500000     0.000000     0.000000      7.000000
50%     967.000000   968.000000     7.000000     1.000000     37.000000
75%    1450.500000  1451.500000    53.500000     6.000000    372.500000
max    1934.000000  1935.000000  5547.000000   975.000000  25922.000000
```

india_covid_19 dataset have total 1935 observations,

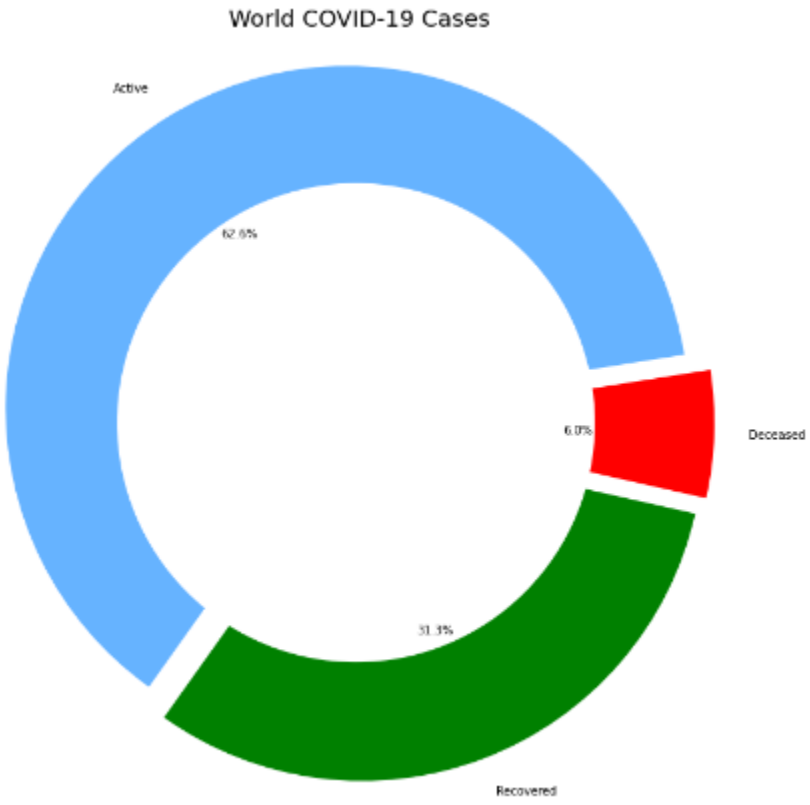
```
 hospital_beds
        Unnamed: 0        Sno  ...  NumUrbanHospitals_NHP18
NumUrbanBeds_NHP18
count   37.000000  37.000000  ...                37.000000
37.000000
mean    18.000000  19.000000  ...               203.891892
23306.648649
std     10.824355  10.824355  ...               616.352568
70502.578529
min      0.000000   1.000000  ...                 0.000000
0.000000
25%      9.000000  10.000000  ...                14.000000
1393.000000
50%     18.000000  19.000000  ...                59.000000
5228.000000
75%     27.000000  28.000000  ...               149.000000
18819.000000
max     36.000000  37.000000  ...              3772.000000
431173.000000
```

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)

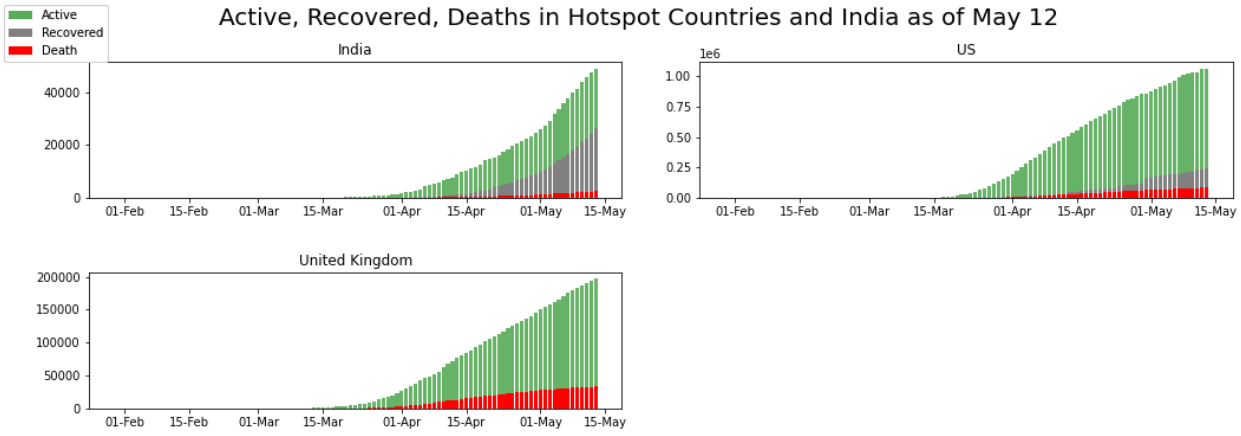


The Above output is the result of world current updates preprocessing code, The results says that there are currently 62.6% of active corona patients, 31.3% of corona patients are recovered from confirmed patients, and 6 % of confirmed patients have deceased.

```
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[cou
ntry]))
    global_active[country] = list(map(int.__sub__, k, global_recovered[cou
ntry]))

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 = 'Dea
th');
    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)



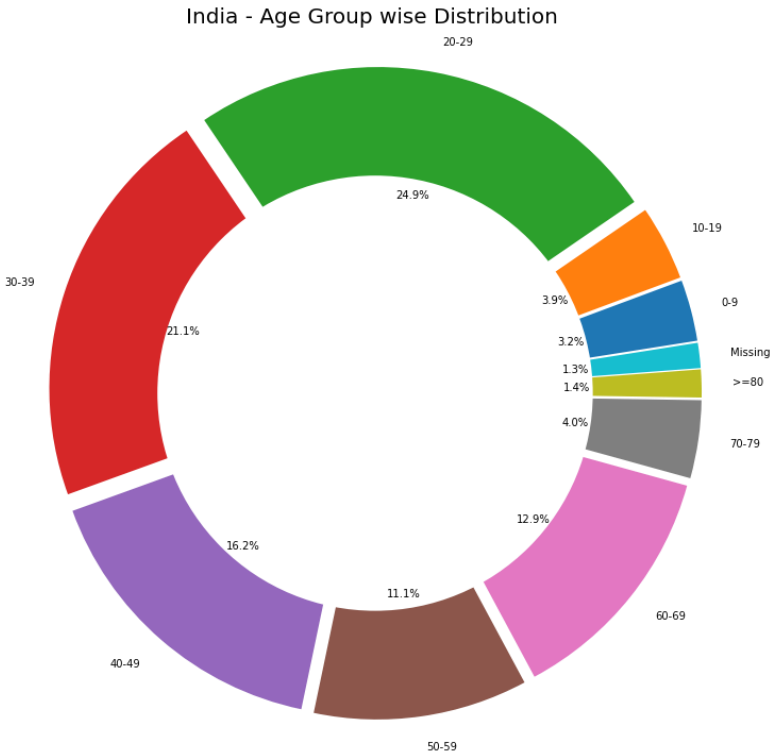
the above plot is self-explanatory, US has the greater number of confirmed corona patients as it has reached above 1 million, In United Kingdom Surprisingly NO data available for recovered corona

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 =explo
de)
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()
```

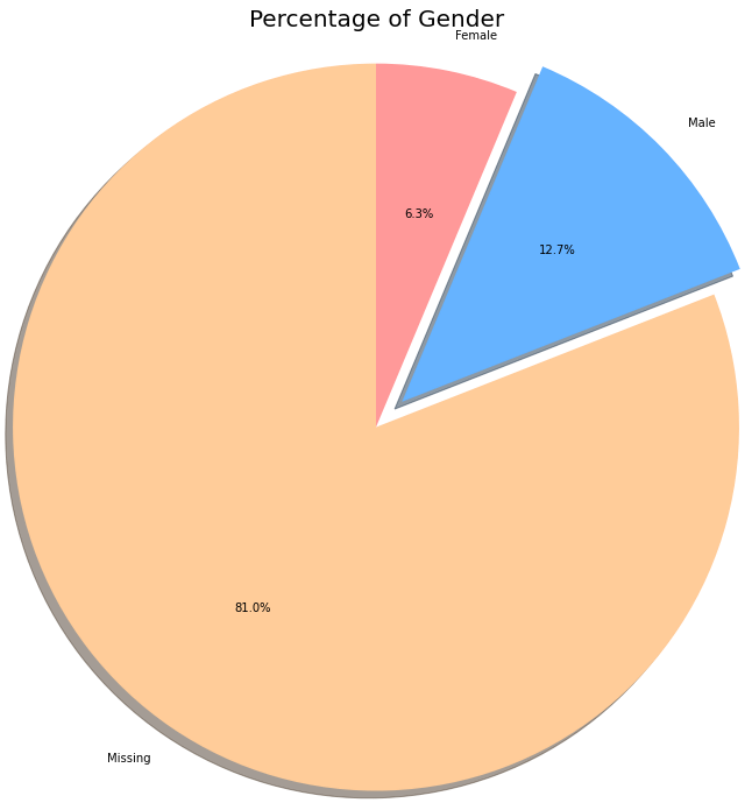
The above line of codes will help to plot the Age Group wise Distribution of India,
The output of the codes is.



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()
```

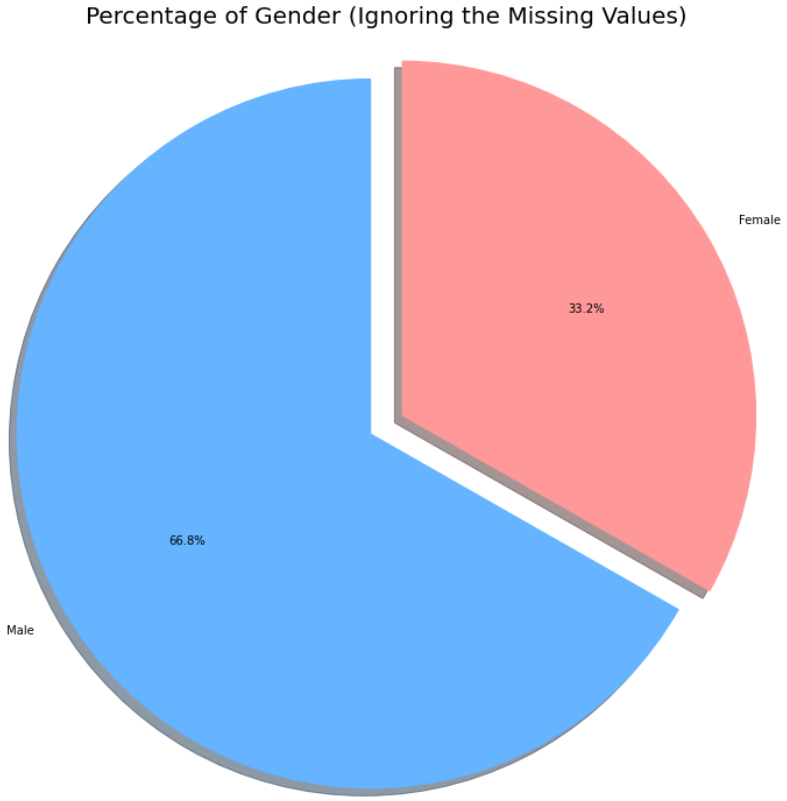
the above will help to create pie chart using ['Missing', 'Male', 'Female'] as labels,



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

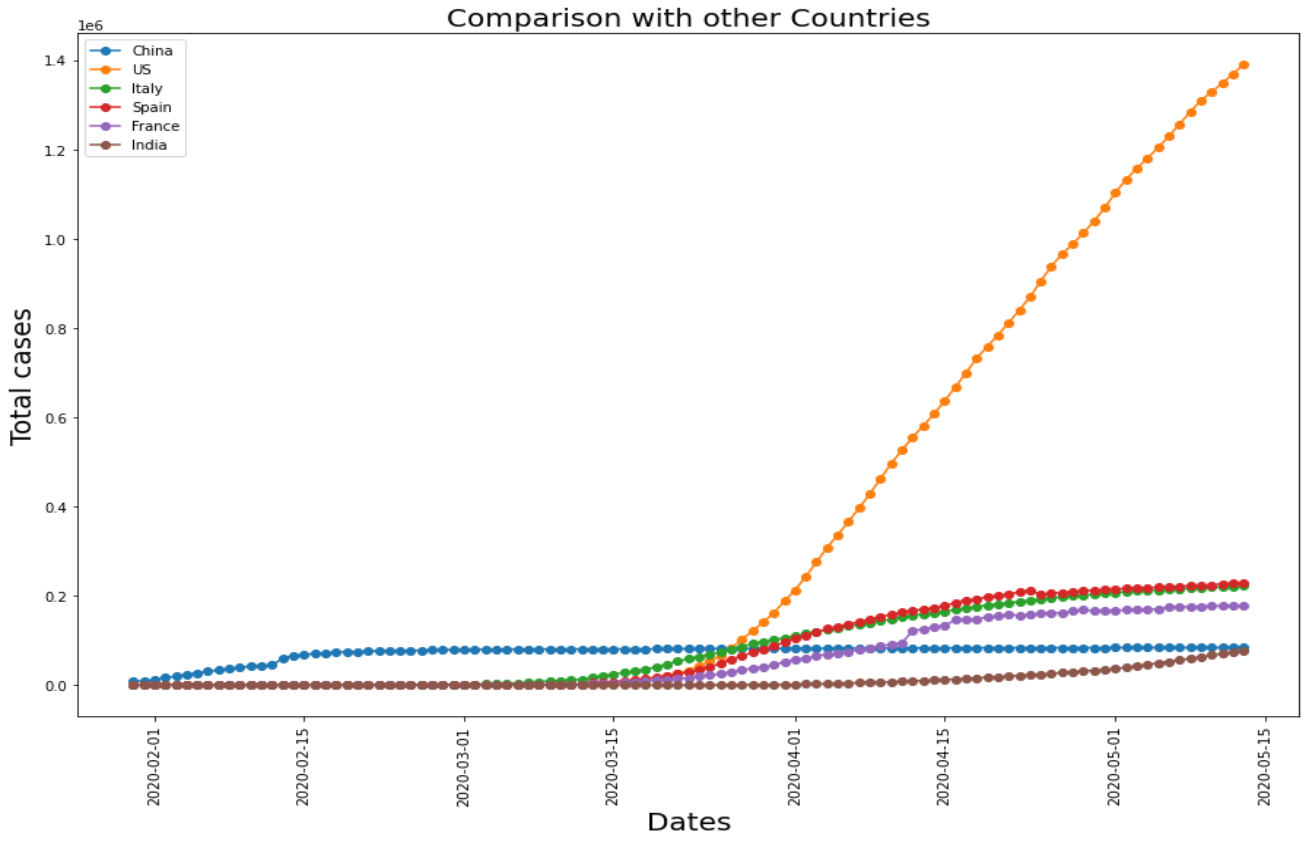
Men are the most affected accounting to 67%, female 33% , But remember we have ~80% data missing.

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();
```

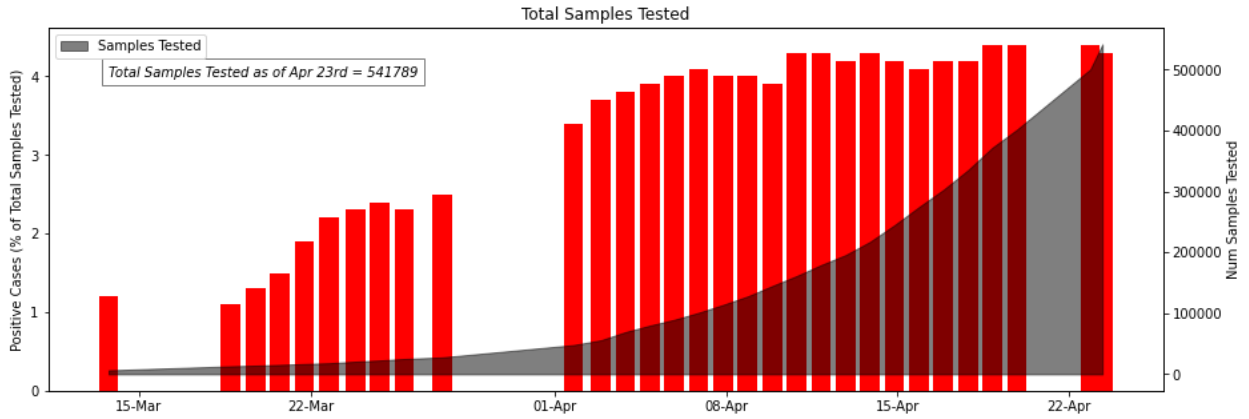
the above code will help to plot a comparison of confirmed cases between 'China','US', 'Italy', 'Spain', 'France','India'



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)
2. Low testing rate (Source: news18)

```
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 (% of 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 23
rd = 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"]/s
tate_cases["Confirmed"],2)
state_cases["Cure Rate (per 100)"] = np.round(100*state_cases["Cured"]/sta
te_cases["Confirmed"],2)
state_cases.sort_values('Confirmed', ascending= False).fillna(0).style.bac
kground_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 R
ate (per 100)"])\
                        .background_gradient(cmap='Blues',subset=["Cure Ra
te (per 100)"])
```

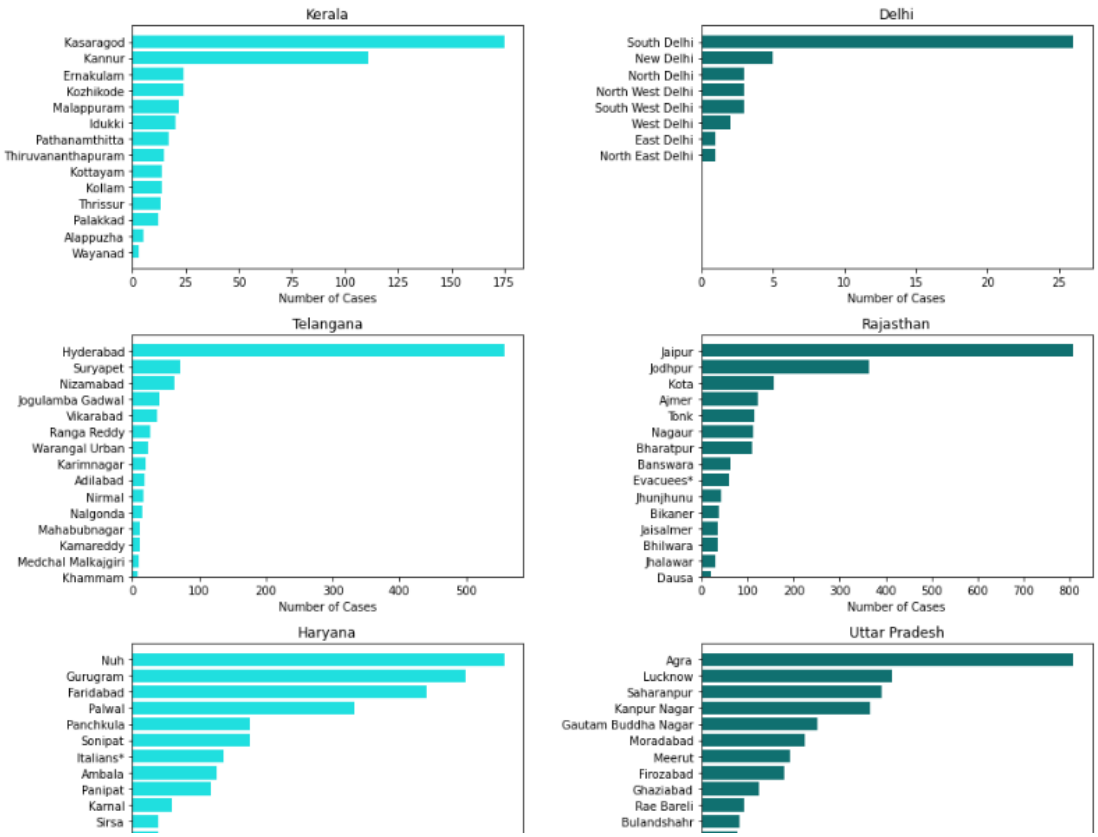
Out[0]:

	State/UnionTerritory	Confirmed	Deaths	Cured	Active	Death Rate (per 100)	Cure Rate (per 100)
20	Maharashtra	25922	975	5547	19400	3.760000	21.400000
10	Gujarat	9267	566	3562	5139	6.110000	38.440000
30	Tamil Nadu	9227	64	2176	6987	0.690000	23.580000
8	Delhi	7998	106	2858	5034	1.330000	35.730000
29	Rajasthan	4328	121	2459	1748	2.800000	56.820000
19	Madhya Pradesh	4173	232	2004	1937	5.560000	48.020000
34	Uttar Pradesh	3729	83	1902	1744	2.230000	51.010000
36	West Bengal	2290	207	702	1381	9.040000	30.660000
1	Andhra Pradesh	2137	47	1142	948	2.200000	53.440000
28	Punjab	1924	32	200	1692	1.660000	10.400000
31	Telengana	1367	34	940	393	2.490000	68.760000
13	Jammu and Kashmir	971	11	466	494	1.130000	47.990000
16	Karnataka	959	33	451	475	3.440000	47.030000
4	Bihar	940	7	388	545	0.740000	41.280000
11	Haryana	793	11	418	364	1.390000	52.710000
26	Odisha	538	3	143	392	0.560000	26.580000
17	Kerala	534	4	490	40	0.750000	91.760000
5	Chandigarh	187	3	28	156	1.600000	14.970000
14	Jharkhand	173	3	79	91	1.730000	45.660000
32	Tripura	155	0	16	139	0.000000	10.320000
3	Assam	80	2	39	39	2.500000	48.750000
33	Unassigned	77	0	0	77	0.000000	0.000000

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']==sta
te]['detected_district'].value_counts().index
    try:
        g = sns.countplot(data=individual_details[individual_details['dete
cted_state']==state],y='detected_district',orient='v',color=colors_list[id
x%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

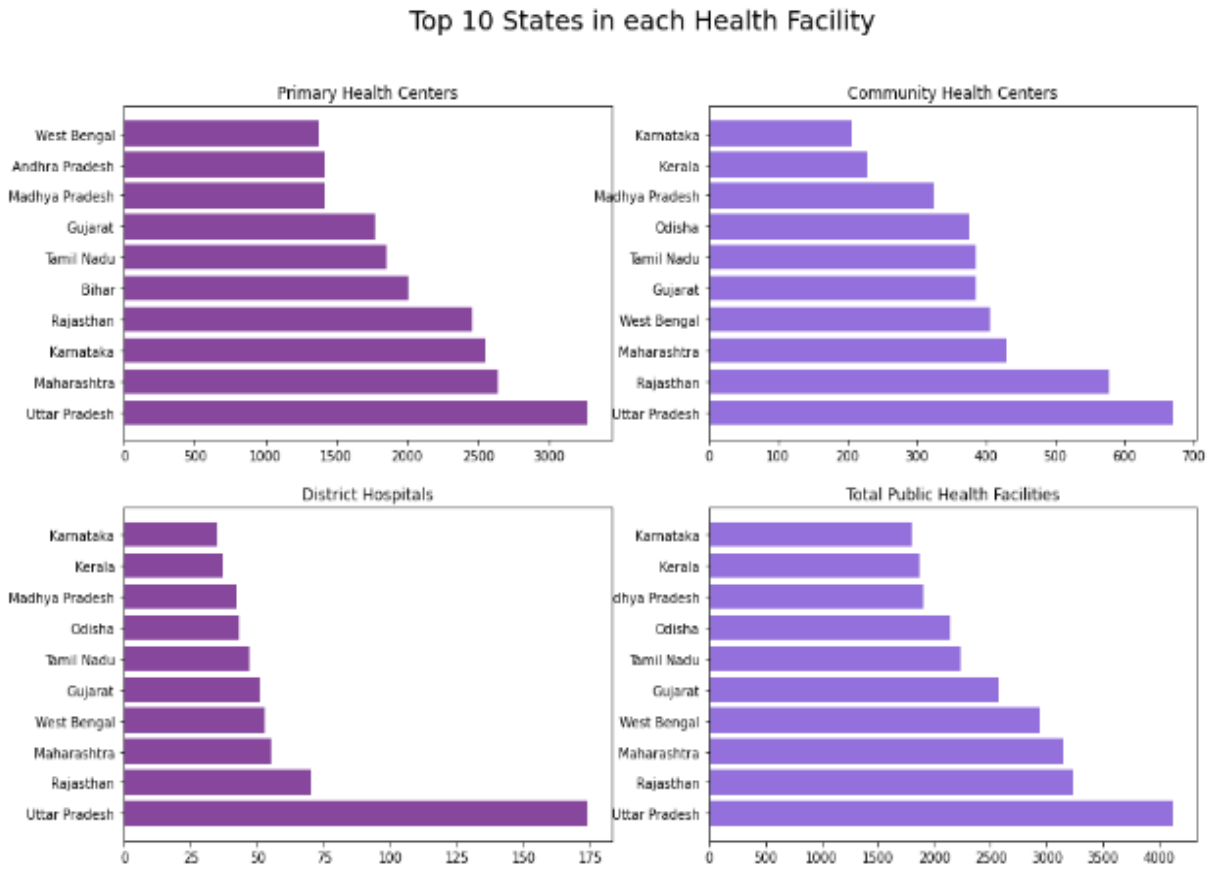


1. Maharashtra- The western state, among the country's most developed, has the largest number of coronavirus patients at 23,264 positive cases including 3,470 discharged patients and 731 deaths.
2. Gujarat – Maharashtra's neighboring state and also among the most industrialized, it has registered 9,723 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_HM
IS')
top_10_district_hospitals = hospital_beds.nlargest(10,'NumDistrictHospital
s_HMIS')
top_10_public_facility = hospital_beds.nlargest(10,'TotalPublicHealthFacil
ities_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['NumPrimaryHealthCenter
s_HMIS'],color ='#87479d');

plt.subplot(222)
plt.title('Community Health Centers')
plt.barh(top_10_community['State/UT'],top_10_community['NumCommunityHealth
Centers_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');
```

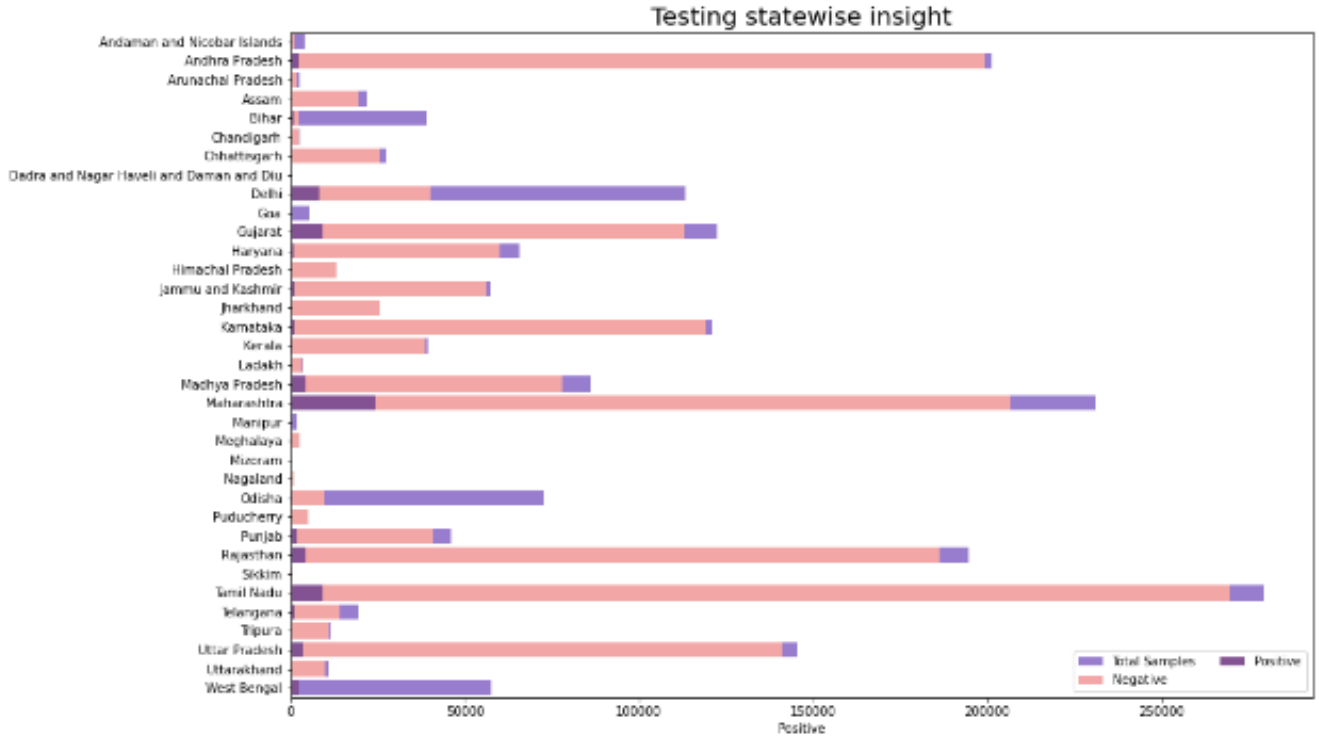


Uttar Pradesh have highest Primary Health Centers, Community Health Centers, Total Public Health Facilities, District Hospitals
Rajasthan is 2nd highest in Community Health Centers, Total Public Health Facilities, District Hospitals

---

```
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')
```

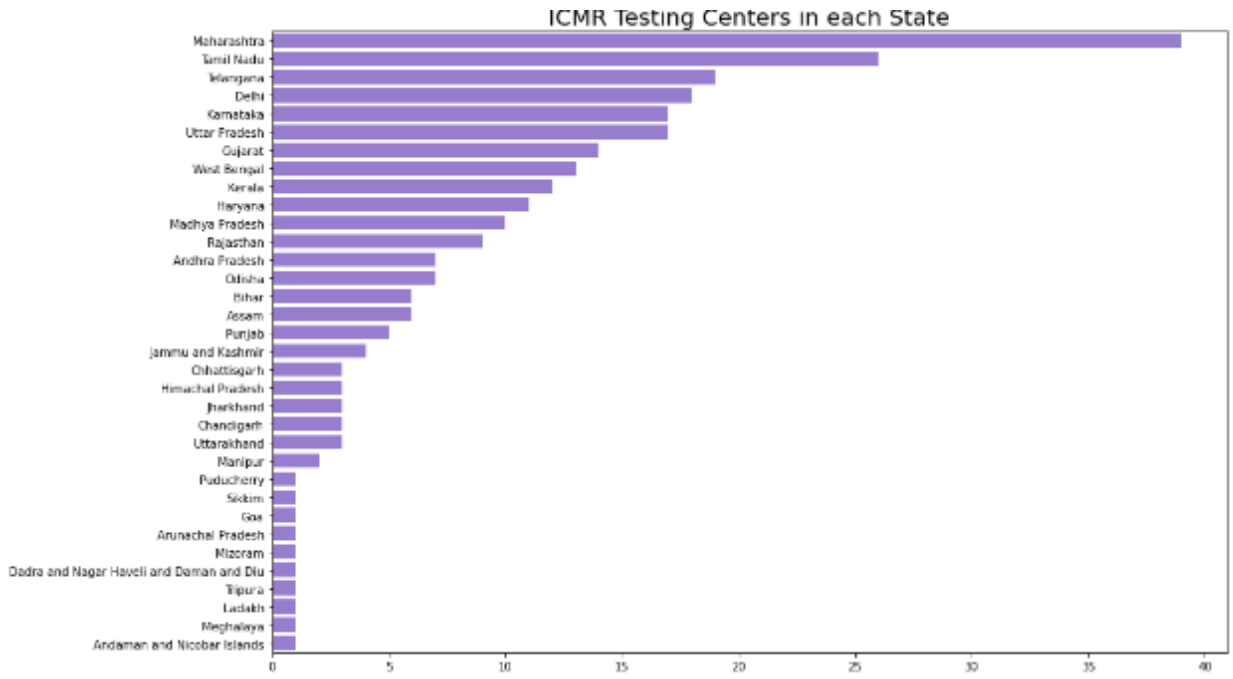
```
sns.barplot(x='Negative', y=state_names, data=state_test,label='Negative',
color= '#ff9999')
sns.barplot(x='Positive', y=state_names, data=state_test,label='Positive',
color='#87479d')
plt.title('Testing statewise insight',fontsize = 20)
plt.legend(ncol=2, loc="lower right", frameon=True);
```



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

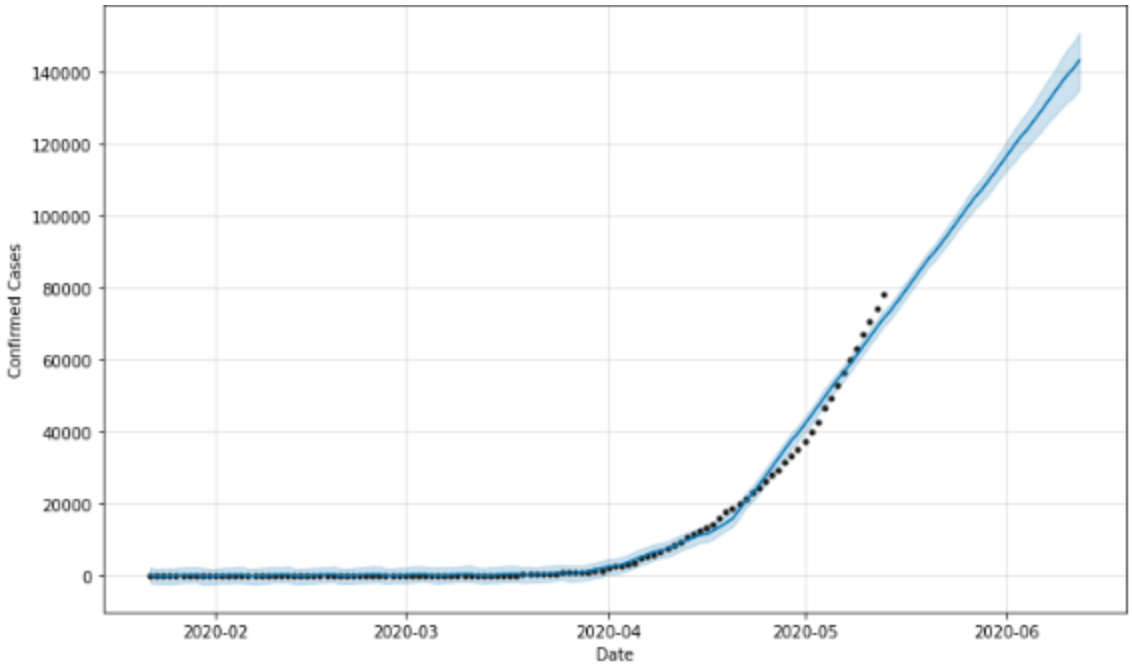
---

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','P
rovince_State', 'Country_Region','ConfirmedCases','Fatalities']
test_columns = ['day','month','dayofweek','dayofyear','quarter','weekofyea
r','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_st
ate=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 c
hange
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_Sta
te,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','d
ayofyear','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]'})
```

	[Mean Squared Error]	[Mean Absolute Error]	[Root Mean Square Error]
LGBM	2041866.06	850807.00	1.645459e+08
Random Forest	473.74	209.52	3.877130e+03
XGBoost	1428.94	922.39	1.282755e+04

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

SQLite

```
1 SELECT * FROM india_covid_19;
```

C2	C3	C4	C5	C6	C7	C8	C9	C10
Sno	Date	Time	State/UnionT...	ConfirmedIndi...	ConfirmedFor...	Cured	Deaths	Confirmed
1	2020-01-30	6:00 PM	Kerala	1	0	0	0	1
2	2020-01-31	6:00 PM	Kerala	1	0	0	0	1
3	2020-02-01	6:00 PM	Kerala	2	0	0	0	2
4	2020-02-02	6:00 PM	Kerala	3	0	0	0	3
5	2020-02-03	6:00 PM	Kerala	3	0	0	0	3
6	2020-02-04	6:00 PM	Kerala	3	0	0	0	3
7	2020-02-05	6:00 PM	Kerala	3	0	0	0	3
8	2020-02-06	6:00 PM	Kerala	3	0	0	0	3
9	2020-02-07	6:00 PM	Kerala	3	0	0	0	3

SQLite

```
1 SELECT AVG(confirmed) FROM india_covid_19;
```

Avg(confirmed)
622.9948320413437

SQLite

```
SELECT state,MAX(cured) AS 'MAXIMUM CURED PER DAY'
FROM india_covid_19
GROUP BY state
```

State	Maximum cured per day
Andaman and Nicobar Islands	33
Andhra Pradesh	975
Arunachal Pradesh	1
Assam	9
Bihar	98
Chandigarh	9
Chhattisgarh	9
Dadar Nagar Haveli	0
Delhi	877

SQLite

```
SELECT state,MAX(deaths) AS 'maximum death per day' FROM india_covid_19
WHERE state IN ('Karnataka', 'Tamil Nadu', 'Telengana','Andhra Pradesh')
GROUP BY state;
```

State	Maximum death per day
Andhra Pradesh	9
Karnataka	9
Tamil Nadu	8
Telengana	9

SQLite

```
SELECT * FROM india_covid_19
WHERE NOT state='Karnataka';
```

Sno	Date	Time	State	Confir...	Confir...	Cured	Deaths	Confirmed
1	1/30/2020	6:00 PM	Kerala	1	0	0	0	1
2	1/31/2020	6:00 PM	Kerala	1	0	0	0	1
3	2/1/2020	6:00 PM	Kerala	2	0	0	0	2
4	2/2/2020	6:00 PM	Kerala	3	0	0	0	3
5	2/3/2020	6:00 PM	Kerala	3	0	0	0	3
6	2/4/2020	6:00 PM	Kerala	3	0	0	0	3
7	2/5/2020	6:00 PM	Kerala	3	0	0	0	3
8	2/6/2020	6:00 PM	Kerala	3	0	0	0	3
9	2/7/2020	6:00 PM	Kerala	3	0	0	0	3

SQLite

```
SELECT COUNT(TIME) AS 'Number_of_times_Checked', state, confirmed
FROM india_covid_19
GROUP BY state
ORDER BY confirmed DESC;
```

Number_of_times_checked	State	Confirmed
72	Uttar Pradesh	6
56	Gujarat	5
3	Unassigned	46
1	Jharkhand#	45
55	Madhya Pradesh	4
50	Goa	3
67	Maharashtra	2
69	Ladakh	2
55	Himachal Pradesh	2

SQLite

```
SELECT state,MAX(confirmedindiannational)
FROM india_covid_19
GROUP BY state
ORDER BY MAX(confirmedindiannational) DESC;
```

State	Max(confirmedindiannational)
West Bengal	9
Uttar Pradesh	9
Kerala	9
Delhi	9
Bihar	9
Andhra Pradesh	9
Maharashtra	86
Telengana	8
Ladakh	8

SQLite

```
SELECT * FROM india_covid_19 WHERE DATE = '5/14/2020' AND state = 'Karnataka' ;
```

Sno	Date	Time	State	Confirmedind...	Confirmedfor...	Cured	Deaths	Confirmed
1918	5/14/2020	8:00 AM	Karnataka	-	-	451	33	959

SQLite

```
1 SELECT * FROM india_covid_19 WHERE state = 'Karnataka' ORDER BY deaths ASC;
```

Sno	Date	Time	State	Confirmedind...	Confirmedfor...	Cured	Deaths	Confirmed
75	3/9/2020	6:00 PM	Karnataka	1	0	0	0	1
90	3/10/2020	6:00 PM	Karnataka	4	0	0	0	4
109	3/11/2020	6:00 PM	Karnataka	4	0	0	0	4
120	3/12/2020	6:00 PM	Karnataka	4	0	0	0	4
133	3/13/2020	6:00 PM	Karnataka	6	0	0	1	6
146	3/14/2020	6:00 PM	Karnataka	6	0	0	1	6
152	3/15/2020	6:00 PM	Karnataka	6	0	0	1	6
166	3/16/2020	6:00 PM	Karnataka	6	0	0	1	6
181	3/17/2020	6:00 PM	Karnataka	11	0	0	1	11

SQLite

Table
- demo
- india_covid_19
  - Column
    - Sno
    - Date
    - Time
    - State
    - ConfirmedIndianNational
    - ConfirmedForeignNational
    - Cured
    - Deaths
    - Confirmed

MariaDB

PostgreSQL

MS SQL

SQLite

```
1 SELECT
2   *,
3   LAG(confirmed) OVER(
4     PARTITION BY state
5     ORDER BY DATE)
6   AS confirmed_previous_day
7 FROM india_covid_19
```

Sno	Date	Time	State	Confi...	Confi...	Cured	Deaths	Confi...	Confirmed_previous_day
1576	5/4/2020	5:00 PM	Andhra Pradesh	-	-	524	36	1650	1583
1608	5/5/2020	5:00 PM	Andhra Pradesh	-	-	589	36	1717	1650
1640	5/6/2020	8:00 AM	Andhra Pradesh	-	-	589	36	1717	1717
1673	5/7/2020	8:00 AM	Andhra Pradesh	-	-	729	36	1777	1717
1706	5/8/2020	8:00 AM	Andhra Pradesh	-	-	780	38	1847	1777
1739	5/9/2020	8:00 AM	Andhra Pradesh	-	-	842	41	1887	1847
804	4/10/2...	5:00 PM	Arunachal Pradesh	-	-	0	0	1	Null
835	4/11/2...	5:00 PM	Arunachal Pradesh	-	-	0	0	1	1
866	4/12/2...	5:00 PM	Arunachal Pradesh	-	-	0	0	1	1

SQLite

```
SELECT state , MAX(Confirmed) AS total_confirmed FROM india_covid_19 GROUP BY state;
```

State	Total_confirmed
Andhra Pradesh	955
Arunachal Pradesh	1
Assam	80
Bihar	96
Chandigarh	94
Chhattisgarh	9
Dadar Nagar Haveli	1
Delhi	97
Goa	7
Gujarat	95

Google drive Link of the video recording

https://drive.google.com/open?id=1wBIxHCoSg9Eu9zTrJCQt_DKCJAqgOCgC

References – source of the dataset downloaded from internet

https://github.com/mohithxoxo/covid19_dataset_may15

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