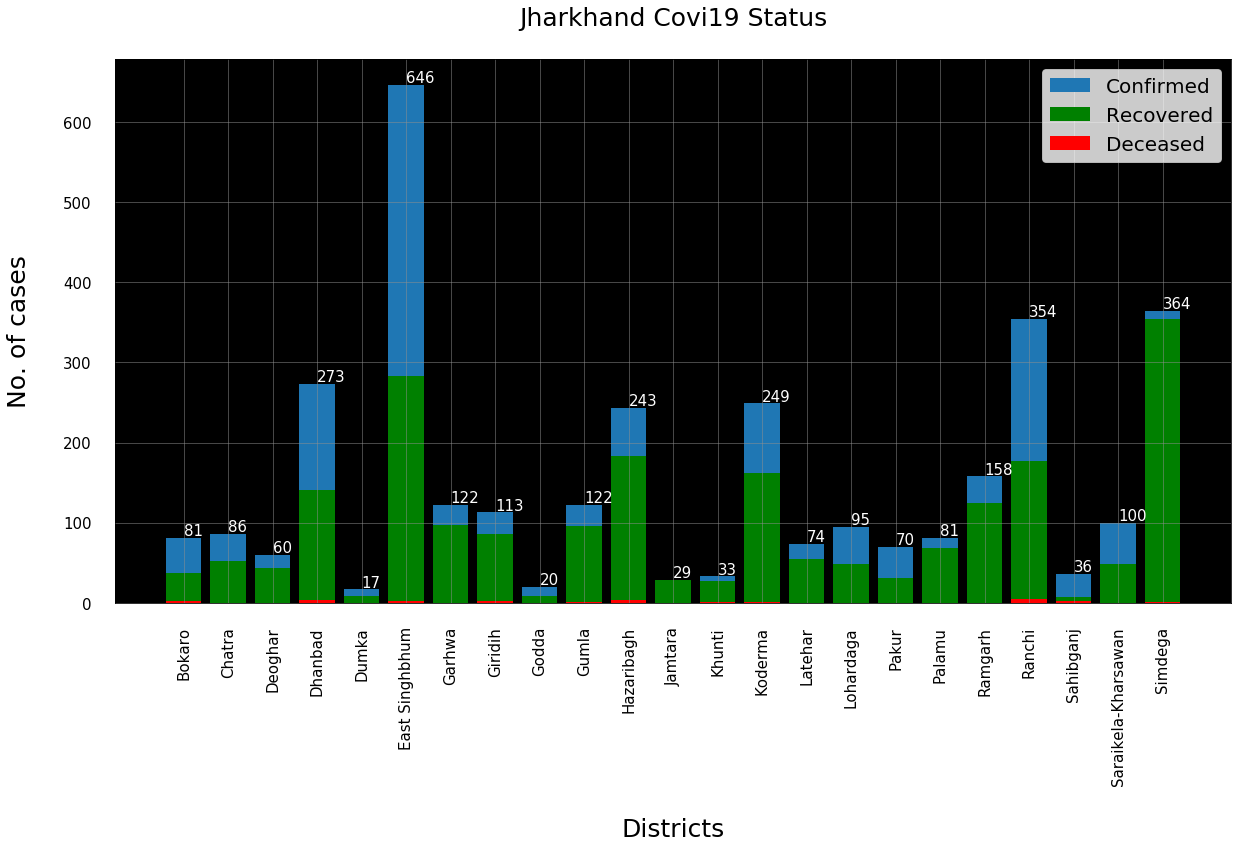
**Major Project of Pandemic Analysis of Covid19**

**By Golmei Shaheamlung**

******

1. ***Present state analytics – India Dataset***

***Code***

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

data = pd.read\_csv('C:/Users/a/Desktop/district.csv')

data.head()

data.info()

re=data.iloc[:23,5].values

de=data.iloc[:23,6].values

co=data.iloc[:23,3].values

x=list(data.iloc[:23,2])

plt.figure(figsize=(20,10))

ax=plt.axes()

ax.set\_facecolor('black')

ax.grid(linewidth=0.5, color='#8f8f8f')

plt.xticks(rotation='vertical', size='15', color='black')#ticks of X

plt.yticks(size='15',color='black')

ax.set\_xlabel('\nDistricts',size=25,

color='black')

ax.set\_ylabel('No. of cases\n',size=25,

color='black')

plt.tick\_params(size=20,color='white')

ax.set\_title('Jharkhand Covi19 Status\n', size=25,color='black')

plt.bar(x,co,label='re')

plt.bar(x,re,label='re',color='green')

plt.bar(x,de,label='re',color='red')

for i,j in zip(x,co):

ax.annotate(str(int(j)),

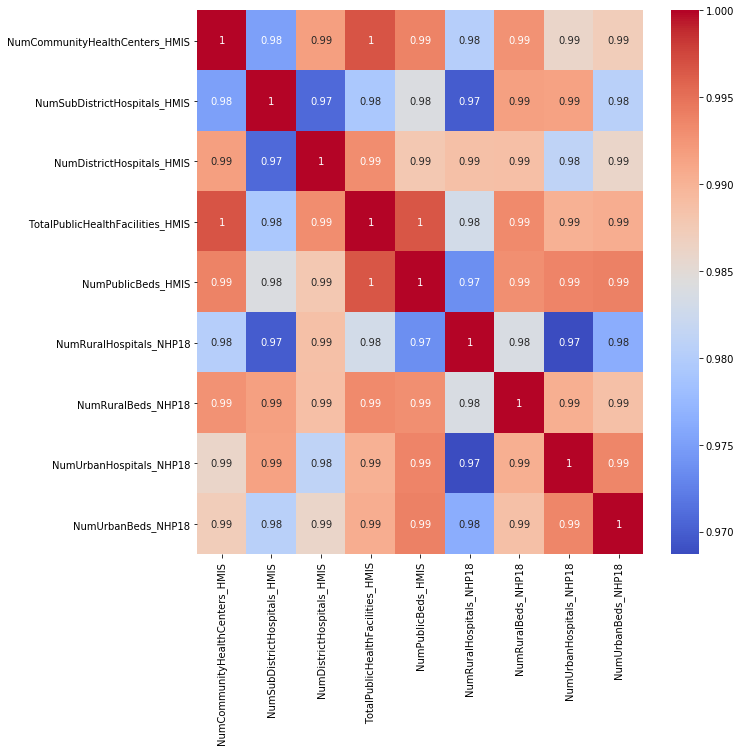
xy=(i,j+3),

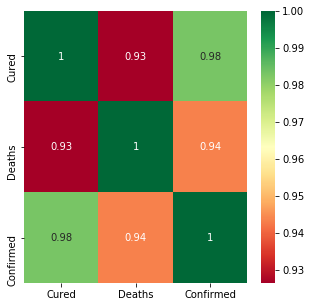
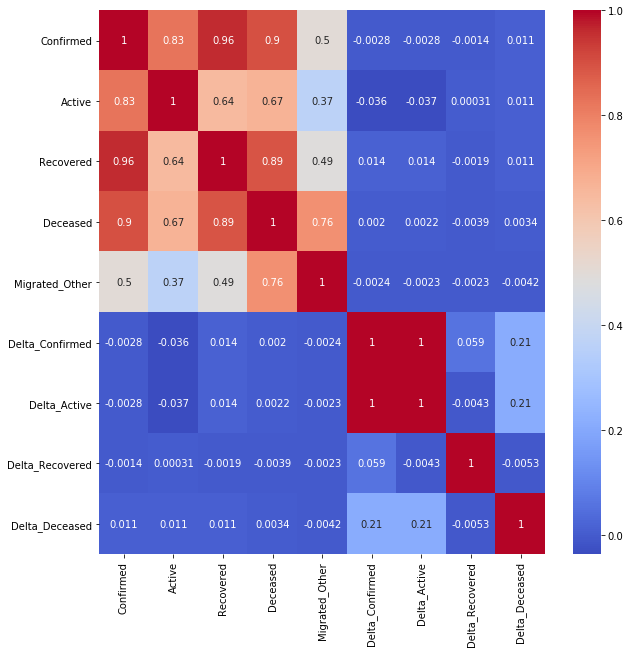
color='white',

size='15')

plt.legend(['Confirmed','Recovered','Deceased'], fontsize=20)

***B.Correlation heatmap for various parameters***

******

******

**Code:**

import pandas as pd

import seaborn

import matplotlib.pyplot as plt

import numpy as np

data=pd.read\_csv("C:/Users/a/Desktop/covid\_19\_india.csv").drop(columns = 'Sno')

data.head()

data.describe()

plt.figure(figsize=(5,5))

seaborn.heatmap(data.corr(),annot=True,cmap="RdYlGn")

district=pd.read\_csv("C:/Users/a/Desktop/district\_wise.csv").drop(columns = 'SlNo').drop(columns = 'Last\_Updated')

district.head()

district.describe()

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

seaborn.heatmap(district.corr(),annot=True,cmap="coolwarm")

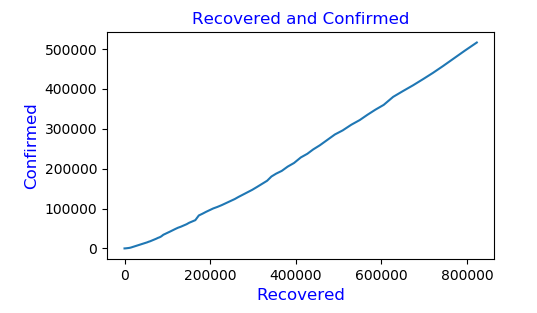
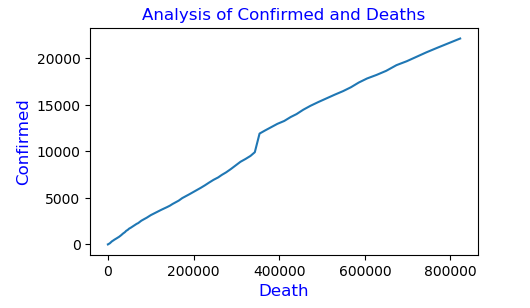
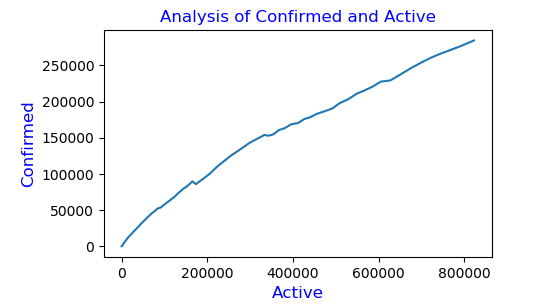
hosp=pd.read\_csv(r"C:/Users/a/Desktop/HospitalBedsIndia.csv").drop(columns='Sno')

hosp.corr()

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

seaborn.heatmap(hospital.corr(),annot=True,cmap="coolwarm")

***C. Build a model for predicting the confirmed and recovered on any dates past 15th July 2020***

Code:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn import linear\_model

import statsmodels.api as sm

df=pd.read\_csv(r"C:/Users/a/Desktop/nation\_level\_daily.csv")

df

df.columns

df.describe()

df.isnull()

reg=linear\_model.LinearRegression()

reg.fit(df[['Confirmed','Deceased','Active']],df.Recovered)

reg.coef\_

reg.intercept\_

##input indepent as Confirmed, deceased,and active

##dependent as recovered will predicted

prediction\_value=reg.predict([[830000,20000,300000]])

print ('Predicted of Recovered : \n', prediction\_value)

##prediction and comparison graph

obj=pd.DataFrame(df,columns=['Date','Confirmed','Recovered','Deceased','Active'])

x=obj['Confirmed']

y=obj['Recovered']

plt.figure(figsize=(5,3),dpi=100)

plt.title('Analysis of Confirmed and Deaths',fontsize=12,color='blue')

plt.xlabel('Recovered',fontsize=12,color='blue')

plt.ylabel('Confirmed',fontsize=12,color='blue')

plt.plot(x,y)

plt.show()

x=obj['Confirmed']

y=obj['Deceased']

plt.figure(figsize=(5,3),dpi=100)

plt.title('Analysis of Confirmed and Deaths',fontsize=12,color='blue')

plt.xlabel('Death',fontsize=12,color='blue')

plt.ylabel('Confirmed',fontsize=12,color='blue')

plt.plot(x,y)

plt.show()

x=obj['Confirmed']

y=obj['Active']

plt.figure(figsize=(5,3),dpi=100)

plt.title('Analysis of Confirmed and Active',fontsize=12,color='blue')

plt.xlabel('Active',fontsize=12,color='blue')

plt.ylabel('Confirmed',fontsize=12,color='blue')

plt.plot(x,y)

plt.show()

***D. Build a classifier for severity gauging***

*import itertools*

*import numpy as np*

*import matplotlib.pyplot as plt*

*from matplotlib.ticker import NullFormatter*

*import pandas as pd*

*import numpy as np*

*import matplotlib.ticker as ticker*

from sklearn import preprocessing

*%matplotlib inline*

*df = pd.read\_csv('C:/Users/a/Desktop/PAML\_COVID\_Dataset.csv')*

*df.head()*

*df.info()*

*df['Patient\_Status '].value\_counts()*

*from sklearn.preprocessing import LabelEncoder*

*label\_encoder = LabelEncoder()*

*df['Patient\_Status '] = label\_encoder.fit\_transform(df['Patient\_Status '])*

*df['Gender ']= label\_encoder.fit\_transform(df['Gender '])*

*df['Serious symptoms']= label\_encoder.fit\_transform(df['Serious symptoms'])*

*df['Less common symptoms']= label\_encoder.fit\_transform(df['Less common symptoms'])*

*df['Most common symptoms']= label\_encoder.fit\_transform(df['Most common symptoms'])*

*df['Severity']= label\_encoder.fit\_transform(df['Severity'])*

*df.columns*

*X = df[['Gender ','Age ','Serious symptoms','Less common symptoms','Most common symptoms','Severity']].values #.astype(float)*

*X[0:5]*

*y = df['Patient\_Status '].values*

*y[0:5]*

*X = preprocessing.StandardScaler().fit(X).transform(X.astype(object))*

*X[0:5]*

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

*X\_train, X\_test, y\_train, y\_test = train\_test\_split( X, y, test\_size=0.2, random\_state=4)*

*print ('Train set:', X\_train.shape, y\_train.shape)*

*print ('Test set:', X\_test.shape, y\_test.shape)*

*from sklearn.neighbors import KNeighborsClassifier*

*k = 4*

*#Train Model and Predict*

*neigh = KNeighborsClassifier(n\_neighbors = k).fit(X\_train,y\_train)*

*neigh*

*yhat = neigh.predict(X\_test)*

*yhat[0:5]*

*from sklearn import metrics*

*print("Train set Accuracy: ", metrics.accuracy\_score(y\_train, neigh.predict(X\_train)))*

*print("Test set Accuracy: ", metrics.accuracy\_score(y\_test, yhat))*

*Ks = 10*

*mean\_acc = np.zeros((Ks-1))*

*std\_acc = np.zeros((Ks-1))*

*ConfustionMx = [];*

*for n in range(1,Ks):*

*#Train Model and Predict*

*neigh = KNeighborsClassifier(n\_neighbors = n).fit(X\_train,y\_train)*

*yhat=neigh.predict(X\_test)*

*mean\_acc[n-1] = metrics.accuracy\_score(y\_test, yhat)*

*std\_acc[n-1]=np.std(yhat==y\_test)/np.sqrt(yhat.shape[0])*

*mean\_acc*

*plt.plot(range(1,Ks),mean\_acc,'g')*

*plt.fill\_between(range(1,Ks),mean\_acc - 1 \* std\_acc,mean\_acc + 1 \* std\_acc, alpha=0.10)*

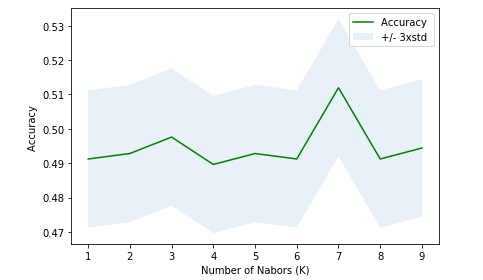
*plt.legend(('Accuracy ', '+/- 3xstd'))*

*plt.ylabel('Accuracy ')*

*plt.xlabel('Number of Nabors (K)')*

*plt.tight\_layout()*

*plt.show()*



Train set Accuracy: 0.6822169059011164

Test set Accuracy: 0.48963317384370014