Demonstration of applying machine learning in COVID related domain

Importing all essential libraries

In [162]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Importing dataset

In [44]:
```
covid = pd.read_csv('Covid_Dataset.csv')

#Displaying dataset with top 5 columns
covid.head(5)
```

Out[44]:

ever	Dry Cough	Sore throat	Running Nose	Asthma	Chronic Lung Disease	Headache	Heart Disease	Diabetes	...	Fatigue	Gastrointestinal	Abroad travel	Contact with COVID Patient	Attended Large Gathering	Vis Pu Expo Pla
Yes	Yes	Yes	Yes	No	No	No	No	Yes	...	Yes	Yes	No	Yes	No	
Yes	Yes	Yes	No	Yes	Yes	Yes	No	No	...	Yes	No	No	No	Yes	
Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	...	Yes	Yes	Yes	No	No	
Yes	Yes	No	No	Yes	No	No	Yes	Yes	...	No	No	Yes	No	Yes	
Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	...	No	Yes	No	Yes	No	

mns

In [45]:
```
#Dimensions of Data
covid.shape
```

Out[45]: (5434, 21)

In [46]:
```
#information about dataset
covid.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5434 entries, 0 to 5433
Data columns (total 21 columns):
 #   Column                                  Non-Null Count  Dtype 
---  ------                                  --------------  ----- 
 0   Breathing Problem                       5434 non-null   object
 1   Fever                                   5434 non-null   object
 2   Dry Cough                               5434 non-null   object
 3   Sore throat                             5434 non-null   object
 4   Running Nose                            5434 non-null   object
 5   Asthma                                  5434 non-null   object
 6   Chronic Lung Disease                    5434 non-null   object
 7   Headache                                5434 non-null   object
 8   Heart Disease                           5434 non-null   object
 9   Diabetes                                5434 non-null   object
 10  Hyper Tension                           5434 non-null   object
 11  Fatigue                                 5434 non-null   object
 12  Gastrointestinal                        5434 non-null   object
 13  Abroad travel                           5434 non-null   object
 14  Contact with COVID Patient              5434 non-null   object
 15  Attended Large Gathering                5434 non-null   object
 16  Visited Public Exposed Places           5434 non-null   object
 17  Family working in Public Exposed Places 5434 non-null   object
 18  Wearing Masks                           5434 non-null   object
 19  Sanitization from Market                5434 non-null   object
 20  COVID-19                                5434 non-null   object
dtypes: object(21)
memory usage: 891.6+ KB
```

In [47]:
```
#summary of the dataset
covid.describe(include='all')
```

Out[47]:

	Fever	Dry Cough	Sore throat	Running Nose	Asthma	Chronic Lung Disease	Headache	Heart Disease	Diabetes	...	Fatigue	Gastrointestinal	Abroad travel	Contact with COVID Patient	Attended Large Gathering	E
5434	5434	5434	5434	5434	5434	5434	5434	5434	...	5434	5434	5434	5434	5434	
2	2	2	2	2	2	2	2	2	...	2	2	2	2	2	
Yes	Yes	Yes	Yes	No	No	Yes	No	No	...	Yes	No	No	Yes	No	
4273	4307	3953	2952	2920	2869	2736	2911	2846	...	2821	2883	2983	2726	2924	

ıs

In [48]:
```
#columns
covid.columns
```

Out[48]:
```
Index(['Breathing Problem', 'Fever', 'Dry Cough', 'Sore throat',
       'Running Nose', 'Asthma', 'Chronic Lung Disease', 'Headache',
       'Heart Disease', 'Diabetes', 'Hyper Tension', 'Fatigue ',
       'Gastrointestinal ', 'Abroad travel', 'Contact with COVID Patient',
       'Attended Large Gathering', 'Visited Public Exposed Places',
       'Family working in Public Exposed Places', 'Wearing Masks',
       'Sanitization from Market', 'COVID-19'],
      dtype='object')
```

In [143]:
```
#to get the unique values from the data
covid.nunique()
```

Out[143]:
```
Breathing Problem                          2
Fever                                      2
Dry Cough                                  2
Sore throat                                2
Running Nose                               2
Asthma                                     2
Chronic Lung Disease                       2
Headache                                   2
Heart Disease                              2
Diabetes                                   2
Hyper Tension                              2
Fatigue                                    2
Gastrointestinal                           2
Abroad travel                              2
Contact with COVID Patient                 2
Attended Large Gathering                   2
Visited Public Exposed Places              2
Family working in Public Exposed Places    2
Wearing Masks                              1
Sanitization from Market                   1
COVID-19                                   2
dtype: int64
```

All the features have uniques values as 'YES' and 'NO' i.e 1 for Yes and 0 for NO. only Wearing Masks and Sanitization from Market have only No as values.

Create a table to get the missing values

In [49]:

```
missing_values=covid.isnull().sum() # missing values

percent_missing = covid.isnull().sum()/covid.shape[0]*100 # missing value %

value = {
    'missing_values ':missing_values,
    'percent_missing %':percent_missing
}
frame=pd.DataFrame(value)
frame
```

Out[49]:

	missing_values	percent_missing %
Breathing Problem	0	0.0
Fever	0	0.0
Dry Cough	0	0.0
Sore throat	0	0.0
Running Nose	0	0.0
Asthma	0	0.0
Chronic Lung Disease	0	0.0
Headache	0	0.0
Heart Disease	0	0.0
Diabetes	0	0.0
Hyper Tension	0	0.0
Fatigue	0	0.0
Gastrointestinal	0	0.0
Abroad travel	0	0.0
Contact with COVID Patient	0	0.0
Attended Large Gathering	0	0.0
Visited Public Exposed Places	0	0.0

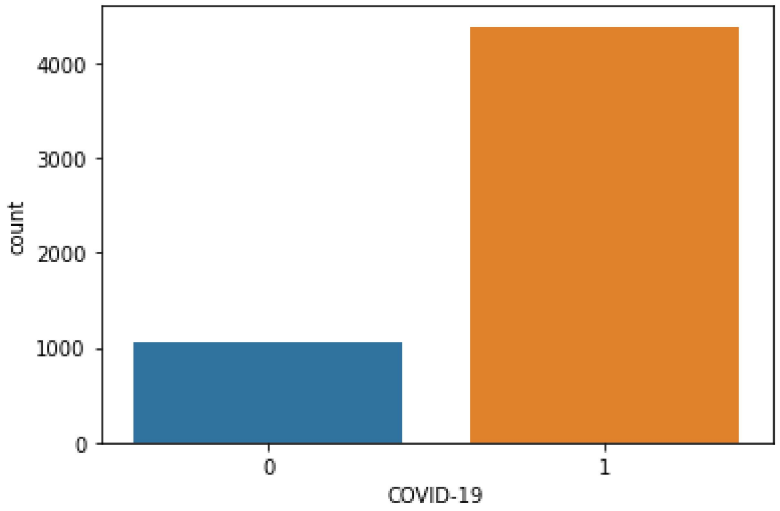
	missing_values	percent_missing %
Family working in Public Exposed Places	0	0.0
Wearing Masks	0	0.0
Sanitization from Market	0	0.0
COVID-19	0	0.0

No Missing Values in the dataset.

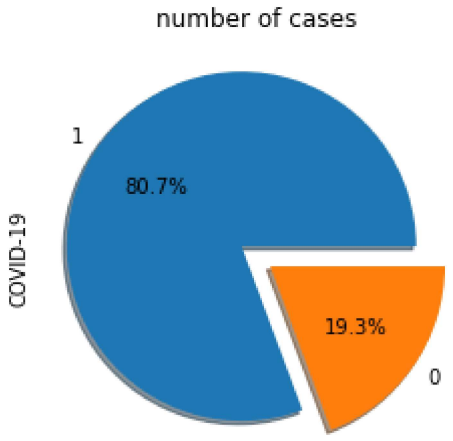
Explorartory data analysis.

In [145]:
```
sns.countplot(x='COVID-19',data=covid)
```

Out[145]:
```
<AxesSubplot:xlabel='COVID-19', ylabel='count'>
```

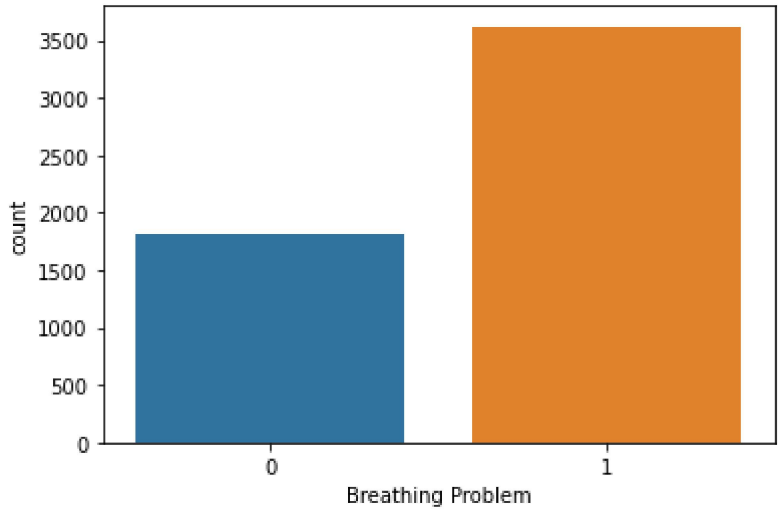


In [146]:
```
covid["COVID-19"].value_counts().plot.pie(explode=[0.1,0.1],autopct='%1.1f%%',shadow=True)
plt.title('number of cases');
```



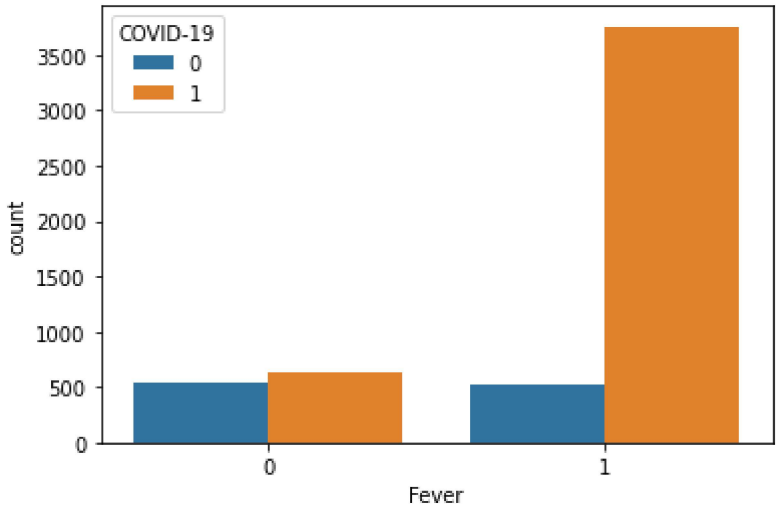
In [147]:
```
sns.countplot(x='Breathing Problem',data=covid)
```

Out[147]: <AxesSubplot:xlabel='Breathing Problem', ylabel='count'>



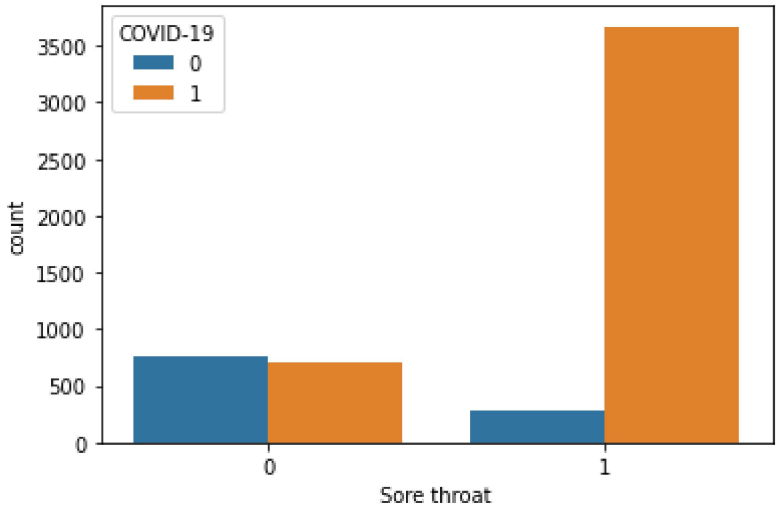
In [148]:
```
sns.countplot(x='Fever',hue='COVID-19',data=covid)
```

Out[148]: <AxesSubplot:xlabel='Fever', ylabel='count'>



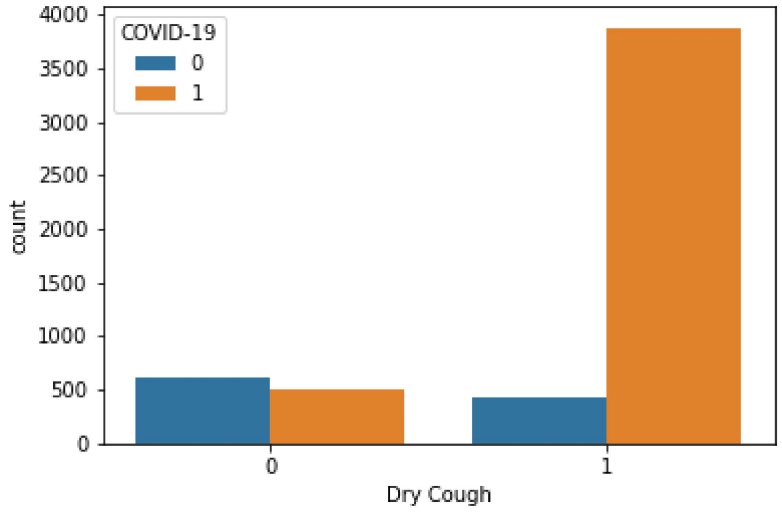
In [149]:
```
sns.countplot(x='Sore throat',hue='COVID-19',data=covid)
```

Out[149]: <AxesSubplot:xlabel='Sore throat', ylabel='count'>



In [150]:
```
sns.countplot(x='Dry Cough',hue='COVID-19',data=covid)
```

Out[150]:
```
<AxesSubplot:xlabel='Dry Cough', ylabel='count'>
```



Visualizing the data

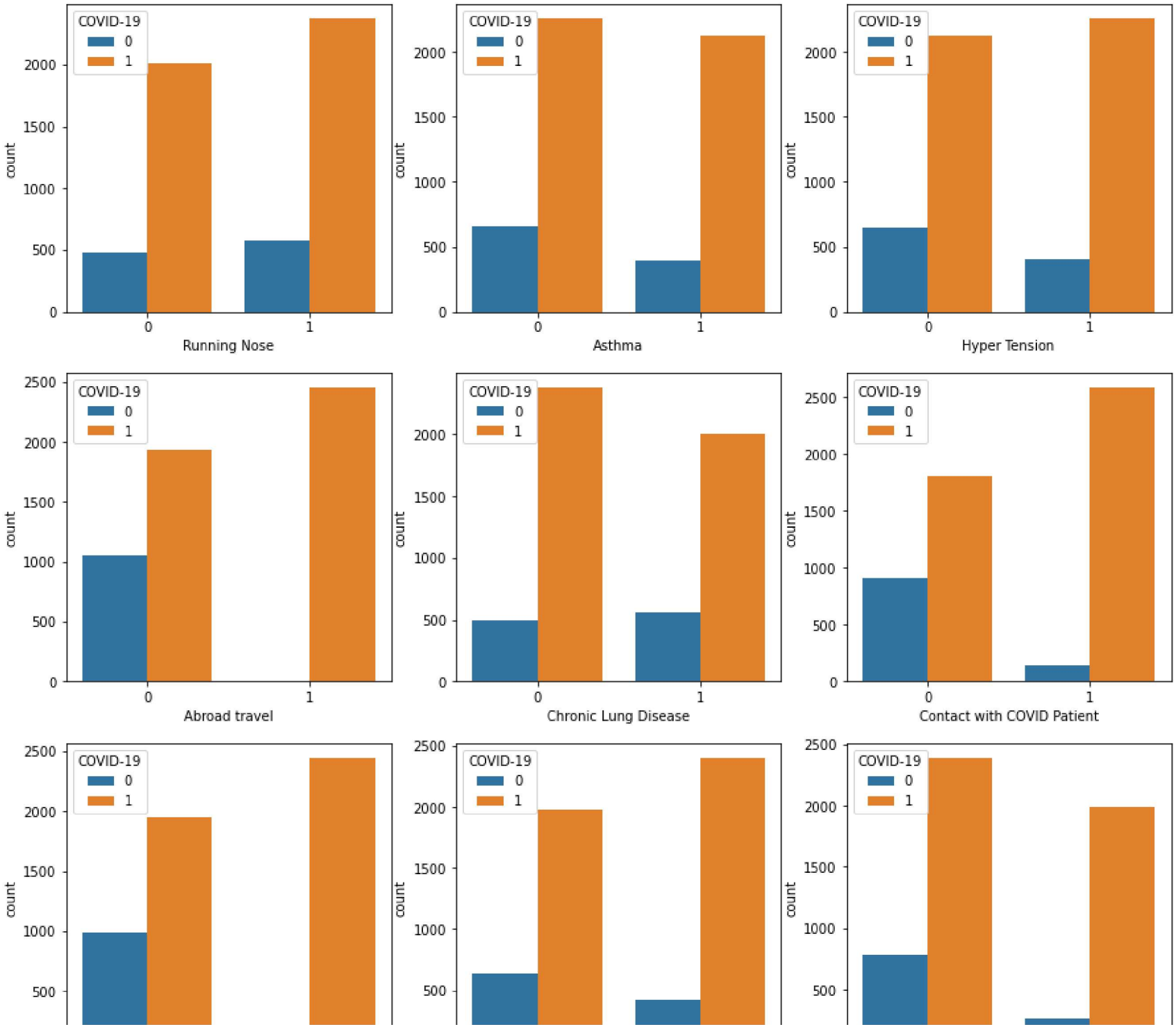
In [152]:
```
feat=['Running Nose','Asthma','Hyper Tension',
      'Abroad travel','Chronic Lung Disease',
      'Contact with COVID Patient','Attended Large Gathering','Visited Public Exposed Places',
      'Family working in Public Exposed Places']
list(enumerate(feat))
```

Out[152]:
```
[(0, 'Running Nose'),
 (1, 'Asthma'),
 (2, 'Hyper Tension'),
 (3, 'Abroad travel'),
 (4, 'Chronic Lung Disease'),
 (5, 'Contact with COVID Patient'),
 (6, 'Attended Large Gathering'),
 (7, 'Visited Public Exposed Places'),
 (8, 'Family working in Public Exposed Places')]
```

In [153]:

```
plt.figure(figsize=(15,30))
for i in enumerate(feat):
    plt.subplot(6,3,i[0]+1)
    sns.countplot(i[1],hue='COVID-19',data=covid)
```

```
C:\Users\asus\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following var
iable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passi
ng other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\asus\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following var
iable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passi
ng other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\asus\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following var
iable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passi
ng other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\asus\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following var
iable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passi
ng other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\asus\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following var
iable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passi
ng other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\asus\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following var
iable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passi
ng other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\asus\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following var
iable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passi
ng other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\asus\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following var
iable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passi
ng other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\asus\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following var
iable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passi
ng other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```


COVID-19
0
1
count
Running Nose
Asthma
Hyper Tension
Abroad travel
Chronic Lung Disease
Contact with COVID Patient

In [ ]:

In [ ]:

Data Preprocessing

In [154]:
```
from sklearn.preprocessing import LabelEncoder
e=LabelEncoder()
```

In [155]:
```
covid['Breathing Problem']=e.fit_transform(covid['Breathing Problem'])
covid['Fever']=e.fit_transform(covid['Fever'])
covid['Dry Cough']=e.fit_transform(covid['Dry Cough'])
covid['Sore throat']=e.fit_transform(covid['Sore throat'])
covid['Running Nose']=e.fit_transform(covid['Running Nose'])
covid['Asthma']=e.fit_transform(covid['Asthma'])
covid['Chronic Lung Disease']=e.fit_transform(covid['Chronic Lung Disease'])
covid['Headache']=e.fit_transform(covid['Headache'])
covid['Heart Disease']=e.fit_transform(covid['Heart Disease'])
covid['Diabetes']=e.fit_transform(covid['Diabetes'])
covid['Hyper Tension']=e.fit_transform(covid['Hyper Tension'])
covid['Abroad travel']=e.fit_transform(covid['Abroad travel'])
covid['Contact with COVID Patient']=e.fit_transform(covid['Contact with COVID Patient'])
covid['Attended Large Gathering']=e.fit_transform(covid['Attended Large Gathering'])
covid['Visited Public Exposed Places']=e.fit_transform(covid['Visited Public Exposed Places'])
covid['Family working in Public Exposed Places']=e.fit_transform(covid['Family working in Public Exposed Pl
covid['Wearing Masks']=e.fit_transform(covid['Wearing Masks'])
covid['Sanitization from Market']=e.fit_transform(covid['Sanitization from Market'])
covid['COVID-19']=e.fit_transform(covid['COVID-19'])
covid['Dry Cough']=e.fit_transform(covid['Dry Cough'])
covid['Sore throat']=e.fit_transform(covid['Sore throat'])
covid['Gastrointestinal ']=e.fit_transform(covid['Gastrointestinal '])
covid['Fatigue ']=e.fit_transform(covid['Fatigue '])
```

In [156]:
```
covid.head()
```

Out[156]:

	Breathing Problem	Fever	Dry Cough	Sore throat	Running Nose	Asthma	Chronic Lung Disease	Headache	Heart Disease	Diabetes	...	Fatigue	Gastrointestinal	Abroad travel	Contac witl COVII Patien
0	1	1	1	1	1	0	0	0	0	1	...	1	1	0	
1	1	1	1	1	0	1	1	1	0	0	...	1	0	0	
2	1	1	1	1	1	1	1	1	0	1	...	1	1	1	
3	1	1	1	0	0	1	0	0	1	1	...	0	0	1	
4	1	1	1	1	1	0	1	1	1	1	...	0	1	0	

5 rows × 21 columns

In [159]:
```
covid.dtypes.value_counts()
```

Out[159]:
```
int64    21
dtype: int64
```

In [160]:
```
covid.describe(include='all')
```

Out[160]:

	Breathing Problem	Fever	Dry Cough	Sore throat	Running Nose	Asthma	Chronic Lung Disease	Headache	Heart Disease	Dial
count	5434.000000	5434.000000	5434.000000	5434.000000	5434.000000	5434.000000	5434.000000	5434.000000	5434.000000	5434.0(
mean	0.666176	0.786345	0.792602	0.727457	0.543246	0.462643	0.472028	0.503497	0.464299	0.47
std	0.471621	0.409924	0.405480	0.445309	0.498172	0.498648	0.499263	0.500034	0.498770	0.49
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0(
25%	0.000000	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0(
50%	1.000000	1.000000	1.000000	1.000000	1.000000	0.000000	0.000000	1.000000	0.000000	0.0(
75%	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.0(
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.0(

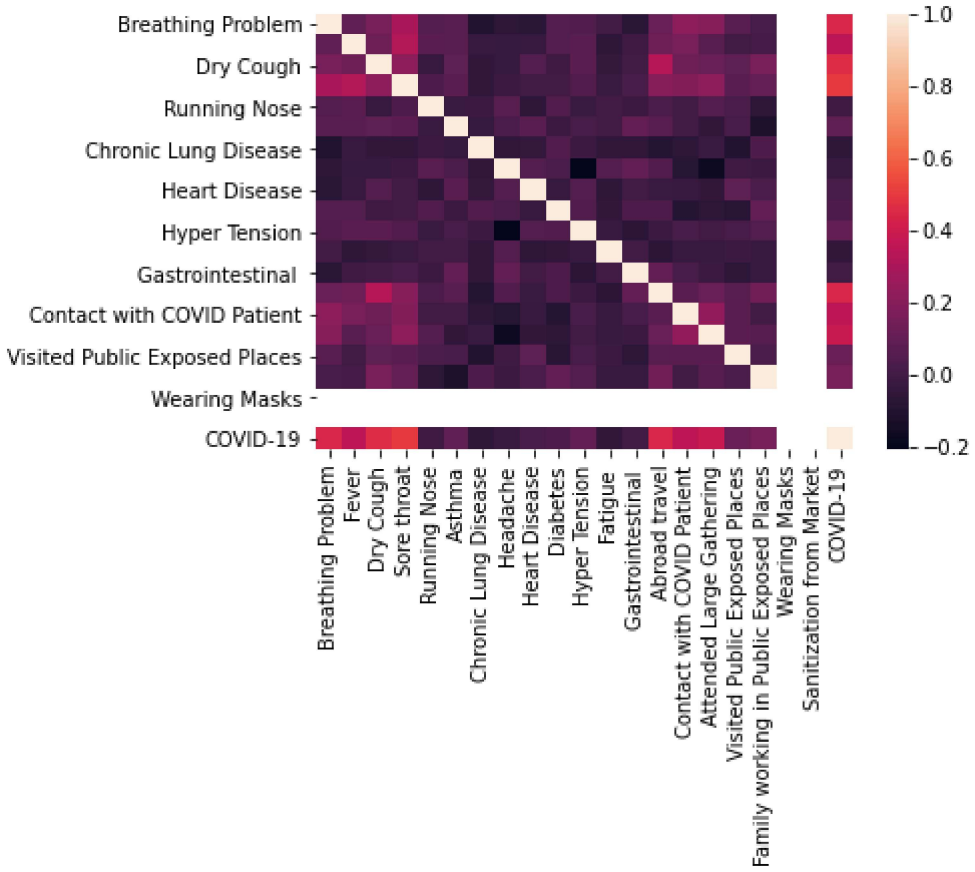
8 rows × 21 columns

In [161]:
```
#Checking null values
covid.isnull().sum()
```

Out[161]:
```
Breathing Problem                          0
Fever                                      0
Dry Cough                                  0
Sore throat                                0
Running Nose                               0
Asthma                                     0
Chronic Lung Disease                       0
Headache                                   0
Heart Disease                              0
Diabetes                                   0
Hyper Tension                              0
Fatigue                                    0
Gastrointestinal                           0
Abroad travel                              0
Contact with COVID Patient                 0
Attended Large Gathering                   0
Visited Public Exposed Places              0
Family working in Public Exposed Places    0
Wearing Masks                              0
Sanitization from Market                   0
COVID-19                                   0
dtype: int64
```

In [168]:
```
sns.heatmap(covid.corr())
```

Out[168]: <AxesSubplot:>



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Splitting Data into Training and Testing set

In [94]:
```
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import accuracy_score
```

In [95]:
```
x=covid.drop('COVID-19',axis=1)
y=covid['COVID-19']
```

In [96]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.20)
```

In [97]:
```
print("x_train size rows and columns :",x_train.shape)
print("x_test size rows and columns :",x_test.shape)
print("x_train size rows and columns :",y_train.shape)
print("x_test size rows and columns :",y_test.shape)
```

```
x_train size rows and columns : (4347, 20)
x_test size rows and columns : (1087, 20)
x_train size rows and columns : (4347,)
x_test size rows and columns : (1087,)
```

In [98]:
```
x_train
```

Out[98]:

	Breathing Problem	Fever	Dry Cough	Sore throat	Running Nose	Asthma	Chronic Lung Disease	Headache	Heart Disease	Diabetes	Hyper Tension	Fatigue	Gastrointestinal	Abroa trave
2939	1	1	0	1	1	1	1	0	0	0	1	0	1	
4201	0	0	0	0	0	0	0	0	1	0	1	1	1	
1286	1	1	1	1	0	1	0	0	0	1	0	1	1	
282	1	1	1	1	0	0	0	0	1	1	1	1	0	
894	1	1	1	1	0	1	0	0	0	1	0	1	1	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
2512	0	1	1	1	0	1	0	0	0	1	0	1	0	
1004	1	1	1	1	1	1	0	1	1	1	0	0	1	
5195	1	0	0	1	1	1	0	1	0	0	1	0	0	
1843	1	1	1	1	0	0	0	0	1	1	1	1	0	
3608	0	1	1	1	0	1	1	0	1	0	0	0	0	

4347 rows × 20 columns

In [99]:
```
x_test
```

Out[99]:

Fever	Dry Cough	Sore throat	Running Nose	Asthma	Chronic Lung Disease	Headache	Heart Disease	Diabetes	Hyper Tension	Fatigue	Gastrointestinal	Abroad travel	Contact with COVID Patient	Atten La Gather
0	1	0	0	0	1	1	0	0	0	1	1	0	0	
1	1	1	1	1	0	0	1	1	1	1	0	1	0	
1	1	1	1	0	1	0	0	1	1	1	0	0	1	
1	0	1	1	0	1	0	0	1	1	0	0	0	1	
1	1	1	1	0	0	0	1	0	0	0	1	0	1	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1	0	1	0	1	0	0	0	1	1	0	0	1	1	
1	1	0	0	0	0	1	0	1	1	1	1	1	1	
1	1	0	0	1	0	0	0	0	1	1	0	1	1	
1	0	1	1	0	1	0	1	1	1	0	1	0	0	
1	1	1	0	0	0	0	1	0	0	0	0	0	0	

columns

In [100]:
```
y_train
```

Out[100]:
```
2939    1
4201    0
1286    1
282     1
894     1
       ..
2512    1
1004    1
5195    1
1843    1
3608    1
Name: COVID-19, Length: 4347, dtype: int32
```

In [101]:
```
y_test
```

Out[101]:
```
4433    0
2210    1
1809    1
5237    1
2033    1
       ..
2992    1
58      1
3530    1
4950    0
2150    1
Name: COVID-19, Length: 1087, dtype: int32
```

In [ ]:
```

```

MODELLING

Logistic Regression

In [115]:
```
#Logistic Regression
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
model = LogisticRegression()
#Fit the model
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
#Score/Accuracy
acc_logreg=model.score(x_test, y_test)*100
print('Accuracy of the model :',acc_logreg)
logreg_confusion=metrics.confusion_matrix(y_test,y_pred)
print('\nConfusion matrix\n:',logreg_confusion)
```

```
Accuracy of the model : 97.33210671573137

Confusion matrix
: [[194  20]
 [  9 864]]
```

K-Nearest Neighbors

In [116]:
```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=20)
knn.fit(x_train, y_train)
y_pred = knn.predict(x_test)
#Score/Accuracy
acc_knn=knn.score(x_test, y_test)*100
print('Accuracy of the model :',acc_knn)
knn_confusion=metrics.confusion_matrix(y_test,y_pred)
print('\nConfusion matrix\n:',knn_confusion)
```

```
Accuracy of the model : 96.78012879484821

Confusion matrix
: [[199  15]
 [ 20 853]]
```

Decision Tree

In [117]:
```
from sklearn import tree
t = tree.DecisionTreeClassifier()
t.fit(x_train,y_train)
y_pred = t.predict(x_test)
#Score/Accuracy
acc_dt=t.score(x_test, y_test)*100
print('Accuracy of the model :',acc_dt)
dt_confusion=metrics.confusion_matrix(y_test,y_pred)
print('\nConfusion matrix\n:',dt_confusion)
```

```
Accuracy of the model : 98.3440662373505

Confusion matrix
: [[209   5]
 [ 13 860]]
```

Naive Bayes

In [118]:
```
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
model.fit(x_train,y_train)
#Score/Accuracy
acc_gnb= model.score(x_test, y_test)*100
print('Accuracy of the model :',acc_gnb)
gnb_confusion=metrics.confusion_matrix(y_test,y_pred)
print('\nConfusion matrix\n:',gnb_confusion)
```

```
Accuracy of the model : 78.1048758049678

Confusion matrix
: [[209   5]
 [ 13 860]]
```

In [119]:
```
# Confusion Matrix

Class_R=metrics.classification_report(y_test,y_pred)
print('\n Classification report:\n',Class_R)
```

```
 Classification report:
               precision    recall  f1-score   support

           0       0.94      0.98      0.96       214
           1       0.99      0.99      0.99       873

    accuracy                           0.98      1087
   macro avg       0.97      0.98      0.97      1087
weighted avg       0.98      0.98      0.98      1087
```

Creating a table for model evaluation

In [123]:
```
models = pd.DataFrame({
    'Model': [ 'K-NearestNeighbor', 'Logistic Regression', 'Naive Bayes',
               'Decision Tree' ],
    'Score': [acc_knn, acc_logreg, acc_gnb, acc_dt]})
models.sort_values(by='Score', ascending=False)
```

Out[123]:

	Model	Score
3	Decision Tree	98.344066
1	Logistic Regression	97.332107
0	K-NearestNeighbor	96.780129
2	Naive Bayes	78.104876

we have got the best accuracy for Decision Tree model followed by K Nearest Neighbors, Logistic Reggression and Support vector Machine and least accuracy with Linear regression an Naive bayes. Here the model gives best accuracy is classification model. Because the problem statement for this dataset is to predict whether the patient comes out to be Covid positive or Negative.

Decision tree is the best suited model with the highest of accuracy amongst all the model classification applied on the problem

statement of this dataset.

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: