1. Introduction

1.1. Importing Libraries

In [1]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

1.2. Importing Dataset

In [2]:
```python
df=pd.read_csv('./COVID-19 Dataset.csv')
df.head()
```

Out[2]:

	test_date	cough	fever	sore_throat	shortness_of_breath	head_ache	corona_result	age_60_and_above	gender	test_indication
0	2020-09-21	0	0	0	0	0	positive	No	male	Other
1	2020-04-07	1	0	0	0	1	positive	No	male	Other
2	2020-08-26	0	0	0	0	0	positive	Yes	female	Contact with confirmed
3	2020-10-14	0	0	0	0	0	positive	No	male	Other
4	2020-09-02	0	0	0	0	0	positive	No	female	Other

In [3]:
```python
df=df.drop(['test_indication','test_date'],axis=1)
df.head()
```

Out[3]:

	cough	fever	sore_throat	shortness_of_breath	head_ache	corona_result	age_60_and_above	gender
0	0	0	0	0	0	positive	No	male
1	1	0	0	0	1	positive	No	male
2	0	0	0	0	0	positive	Yes	female
3	0	0	0	0	0	positive	No	male
4	0	0	0	0	0	positive	No	female

In [4]:
```python
df.describe()
```

Out[4]:

	cough	fever	sore_throat	shortness_of_breath	head_ache
count	400000.000000	400000.000000	400000.000000	400000.000000	400000.000000
mean	0.118572	0.134365	0.052840	0.019290	0.107095
std	0.323285	0.341044	0.223714	0.137543	0.309234
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000

In [5]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400000 entries, 0 to 399999
Data columns (total 8 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   cough                400000 non-null  int64
 1   fever                400000 non-null  int64
 2   sore_throat          400000 non-null  int64
 3   shortness_of_breath  400000 non-null  int64
 4   head_ache            400000 non-null  int64
 5   corona_result        400000 non-null  object
 6   age_60_and_above     350088 non-null  object
 7   gender               392248 non-null  object
dtypes: int64(5), object(3)
memory usage: 24.4+ MB
```

In [6]:
```python
print("shape:", df.shape)
```

```
shape: (400000, 8)
```

1.3. Missing Values

In [7]:
```python
df.isnull().sum()
```

Out[7]:
```
cough                      0
fever                      0
sore_throat                0
shortness_of_breath        0
head_ache                  0
corona_result              0
age_60_and_above       49912
gender                  7752
dtype: int64
```

In [8]:
```python
df=df.dropna()
print("shape:", df.shape)
```

```
shape: (349038, 8)
```

1.4. Convert string to number

In [9]:
```python
df['age_60_and_above'] = df['age_60_and_above'].replace(['No','Yes'],[0,1])
df['corona_result'] = df['corona_result'].replace(['negative','positive'],[0,1])
df['gender'] = df['gender'].replace(['female','male'],[0,1])
```

In [10]:
```python
df.head()
```

Out[10]:

	cough	fever	sore_throat	shortness_of_breath	head_ache	corona_result	age_60_and_above	gender
0	0	0	0	0	0	1	0	1
1	1	0	0	0	1	1	0	1
2	0	0	0	0	0	1	1	0
3	0	0	0	0	0	1	0	1
4	0	0	0	0	0	1	0	0

In [11]:
```python
df.describe()
```

Out[11]:

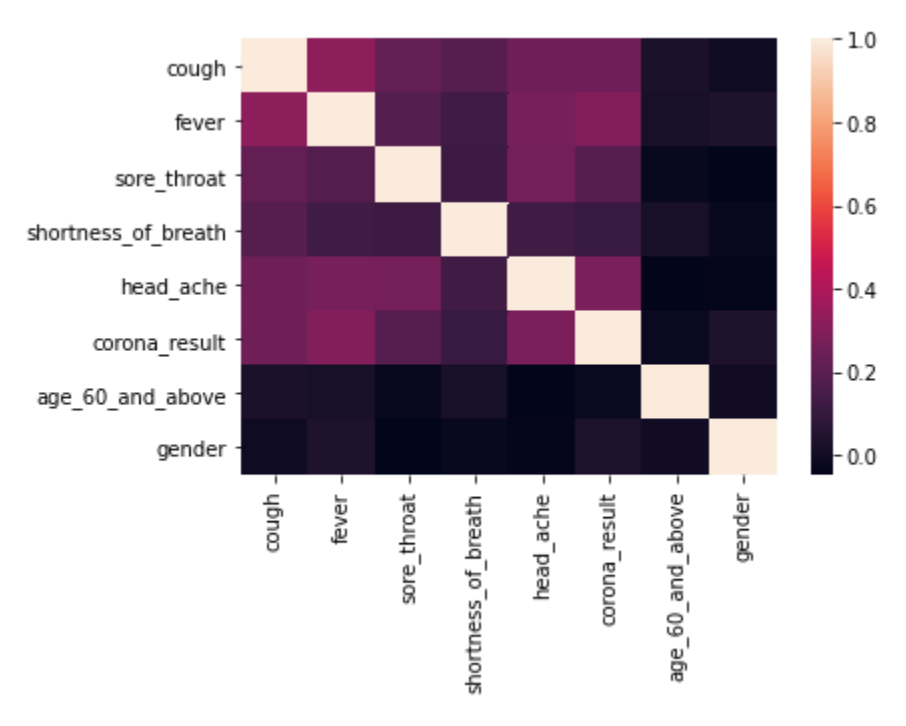
	cough	fever	sore_throat	shortness_of_breath	head_ache	corona_result	age_60_and_above	gender
count	349038.000000	349038.000000	349038.000000	349038.000000	349038.000000	349038.000000	349038.000000	349038.000000
mean	0.124147	0.143523	0.056779	0.020204	0.116219	0.598004	0.126685	0.503286
std	0.329750	0.350606	0.231420	0.140698	0.320488	0.490302	0.332621	0.499990
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	1.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	1.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

2. Data Visualization

2.1 Corr Heat Map

In [12]:
```python
corr = df.corr()
sns.heatmap(corr)
```

Out[12]:
```
<AxesSubplot:>
```



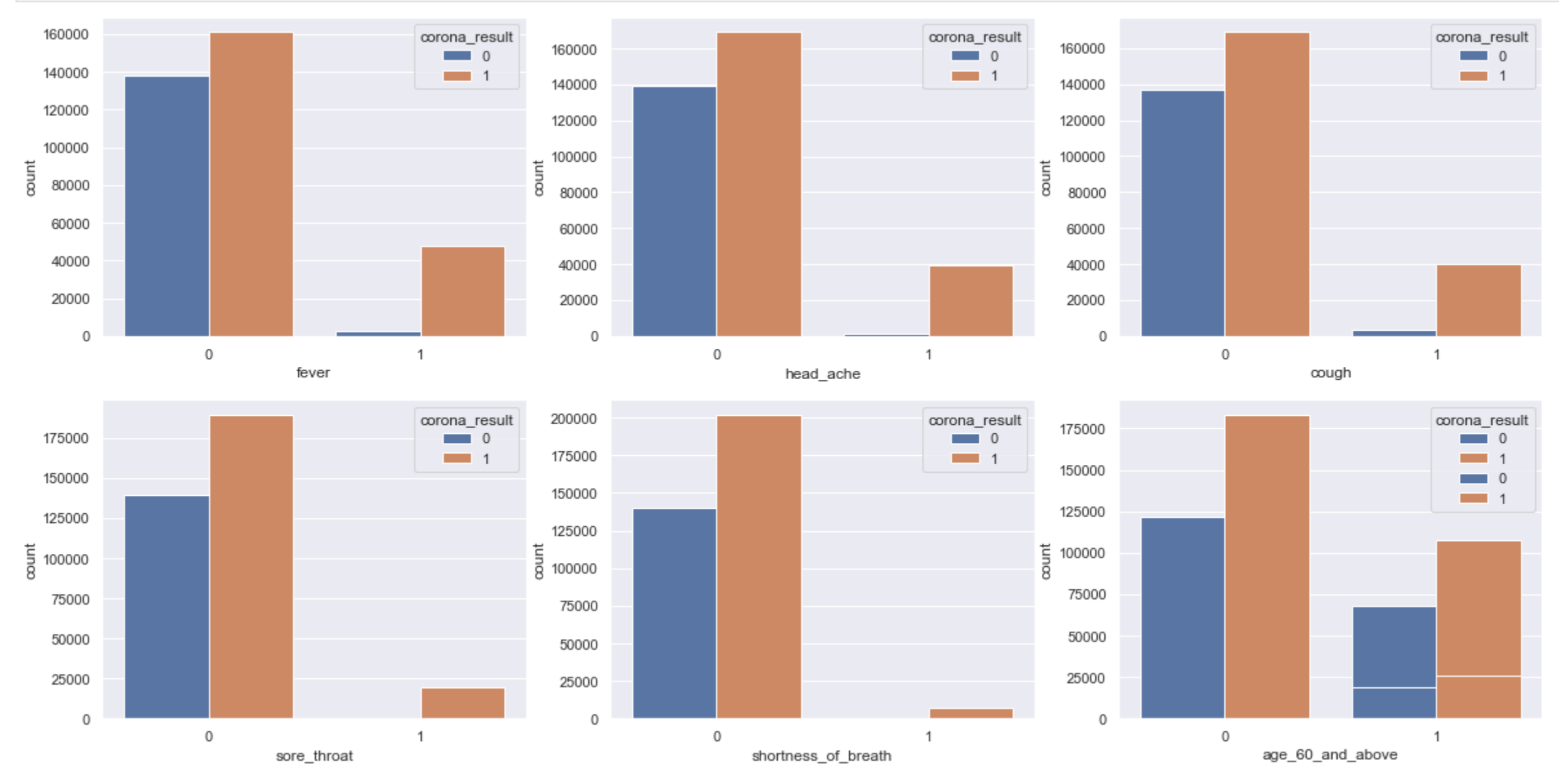
In [13]:
```python
corr.sort_values(by=['corona_result'],ascending=False).iloc[0].sort_values(ascending=False)
```

Out[13]:
```
corona_result          1.000000
fever                  0.297365
head_ache              0.279854
cough                  0.248007
sore_throat            0.189269
shortness_of_breath    0.111050
gender                 0.031731
age_60_and_above      -0.015416
Name: corona_result, dtype: float64
```

In [14]:
```python
# maybe delete gender and age
```

2.2 Count Plot

In [15]:
```python
plt.figure(figsize=(20,10))
sns.set_theme(style='darkgrid')
plt.subplot(2,3,1)
sns.countplot(data=df,x='fever', hue='corona_result')
plt.subplot(2,3,2)
sns.countplot(data=df,x='head_ache', hue='corona_result')
plt.subplot(2,3,3)
sns.countplot(data=df,x='cough', hue='corona_result')
plt.subplot(2,3,4)
sns.countplot(data=df,x='sore_throat', hue='corona_result')
plt.subplot(2,3,5)
sns.countplot(data=df,x='shortness_of_breath', hue='corona_result')
plt.subplot(2,3,6)
sns.countplot(data=df,x='gender', hue='corona_result')
plt.subplot(2,3,6)
sns.countplot(data=df,x='age_60_and_above', hue='corona_result')
plt.show()
```



3. Modeling

3.1. Train & Test Data

In [16]:
```python
X = df.drop(['corona_result'],axis=1)
Y = df[['corona_result']]
```

In [17]:
```python
print('X Shape', X.shape)
print('Y Shape',Y.shape)
```

```
X Shape (349038, 7)
Y Shape (349038, 1)
```

In [18]:
```python
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=42)

print('Number of x_train df',X_train.shape)
print('Number of x_test df',X_test.shape)
print('Number of Y_train df',Y_train.shape)
print('Number of Y_test df',Y_test.shape)
```

```
Number of x_train df (244326, 7)
Number of x_test df (104712, 7)
Number of Y_train df (244326, 1)
Number of Y_test df (104712, 1)
```

3.2. Import Models

In [42]:
```python
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB,BernoulliNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
```

3.3. Model evaluation

In [20]:
```python
models = []
models.append(['Logistic Regression',LogisticRegression(random_state=0)])
models.append(['GaussianNB',GaussianNB()])
models.append(['BernoulliNB',BernoulliNB()])
models.append(['KNeigbors',KNeighborsClassifier()])
models.append(['DecisionTree',DecisionTreeClassifier(random_state=0)])
models.append(['RandomForest',RandomForestClassifier(random_state=0)])
```

In [84]:
```python
from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_score
lst_1 = []
for m in range(len(models)):
    lst_2 = []
    model = models[m][1]
    model.fit(X_train,Y_train)
    Y_pred = model.predict(X_test)
    accuracy = accuracy_score(Y_test,Y_pred)
    precision = precision_score(Y_test,Y_pred)
    recall = recall_score(Y_test,Y_pred)
    f1 = f1_score(Y_test,Y_pred)
    print(models[m][0],':')
    print('Accuracy Score: ',accuracy)
    print('')
    print('Precision: {:.2f} %'.format(precision))
    print('')
    print('Recall: {:.2f} %'.format(recall))
    print('')
    print('F1 Score: {:.2f} %'.format(f1))
    print('-'*40)
    print('')
    lst_2.append(models[m][0])
    lst_2.append(accuracy)
    lst_2.append(precision)
    lst_2.append(recall)
    lst_2.append(f1)
    lst_2.append(models[m][1])
    lst_1.append(lst_2)
```

```
Logistic Regression :
Accuracy Score:  0.6474807089922836

Precision: 0.95 %

Recall: 0.43 %

F1 Score: 0.60 %
----------------------------------------

GaussianNB :
Accuracy Score:  0.6474807089922836

Precision: 0.95 %

Recall: 0.43 %

F1 Score: 0.60 %
----------------------------------------

BernoulliNB :
Accuracy Score:  0.6474807089922836

Precision: 0.95 %

Recall: 0.43 %

F1 Score: 0.60 %
----------------------------------------

KNeigbors :
Accuracy Score:  0.6307109022843609

Precision: 0.67 %

Recall: 0.77 %

F1 Score: 0.71 %
----------------------------------------

DecisionTree :
Accuracy Score:  0.6474807089922836

Precision: 0.95 %

Recall: 0.43 %

F1 Score: 0.60 %
----------------------------------------

RandomForest :
Accuracy Score:  0.6474807089922836

Precision: 0.95 %

Recall: 0.43 %

F1 Score: 0.60 %
----------------------------------------
```

In [85]:
```python
lst_2=[]
model = LinearRegression()
model.fit(X_train, Y_train)
Y_pred = model.predict(X_test)
for i in range(len(Y_pred)):
    if(Y_pred[i][0]>0.5):
        Y_pred[i][0]=1
    else:
        Y_pred[i][0]=0
accuracy = accuracy_score(Y_test,Y_pred)
precision = precision_score(Y_test,Y_pred)
recall = recall_score(Y_test,Y_pred)
f1 = f1_score(Y_test,Y_pred)
print('Accuracy Score: ',accuracy)
print('')
print('Precision: {:.2f} %'.format(precision))
print('')
print('Recall: {:.2f} %'.format(recall))
print('')
print('F1 Score: {:.2f} %'.format(f1))
print('-'*40)
print('')
lst_2.append("Linear Regression")
lst_2.append(accuracy)
lst_2.append(precision)
lst_2.append(recall)
lst_2.append(f1)
lst_2.append(model)
lst_1.append(lst_2)
```

```
Accuracy Score:  0.6414451065780427

Precision: 0.70 %

Recall: 0.70 %

F1 Score: 0.70 %
----------------------------------------
```

In [98]:
```python
lst_show=[]
for i in range(len(lst_1)):
    lst_show.append(lst_1[i][:5])
lst_show
df2 = pd.DataFrame(lst_show,columns=['Model','Accuracy','Precision','Recall','F1 Score'])

df2.sort_values(by=['Accuracy','F1 Score'],inplace=True,ascending=False)
df2
```

Out[98]:

	Model	Accuracy	Precision	Recall	F1 Score
0	Logistic Regression	0.647481	0.949101	0.434880	0.596460
1	GaussianNB	0.647481	0.949101	0.434880	0.596460
2	BernoulliNB	0.647481	0.949101	0.434880	0.596460
4	DecisionTree	0.647481	0.949101	0.434880	0.596460
5	RandomForest	0.647481	0.949101	0.434880	0.596460
6	Linear Regression	0.641445	0.701303	0.699346	0.700323
3	KNeigbors	0.630711	0.666884	0.766380	0.713178

4. input & output

In [114…
```python
from pandas import DataFrame

# fever = 1
# cought = 1
# sore_throat = 1
# shortness_of_breath = 0
# head_ach = 1
# age_60_or_above = 0
# gender = 0

data = [[1,1,1,0,1,0,0]]
col = ['fever','cought','sore_throat','shortness_of_breath','head_ach','age_60_or_above','gender']
test = DataFrame(data, columns=col)
for i in range(len(lst_1)):
    model = lst_1[i][5]
    print(lst_1[i][0],"-->","result:",model.predict(test))
```

```
Logistic Regression --> result: [1]
GaussianNB --> result: [1]
BernoulliNB --> result: [1]
KNeigbors --> result: [1]
DecisionTree --> result: [1]
RandomForest --> result: [1]
Linear Regression --> result: [[1.35751992]]
```