Name: Darshan Rajopadhye
Roll No.: 17CO045
Subject: Machine Learning
Class: BE Computer 1

Assignment 3:
Problem Statement:

Search a COVID related dataset
1. Download the dataset
2. Perform Pre-processing on the dataset
3. Use this dataset to build a Naïve Bayes Classifier
4. Use this dataset to build a Decision Tree Classifier
5. Apply Ensemble to improve the accuracy.

Tool used:
Jupyter Notebook(Python).

Dataset:
The dataset consists of the symptoms of over 2 lakh people from all over the world classified as positive or negative.

Implementation:
Below attached is the converted notebook file with output.

Conclusion:
Test data set performance of both the models are:

1. Naive Bayes: 83.0%
2. Decision Tree:83.3%
3. Gradient Boosting: 86.27%
4. Random Forest:83.58%

Importing libraries

In [62]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sn
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
from sklearn.naive_bayes import GaussianNB
```

Reading dataset

In [28]:

```
df=pd.read_csv("covid_symptoms.csv")
df
```

Out[28]:

	cough	fever	sore_throat	shortness_of_breath	head_ache	corona_result	age_60_and_above	gen
0	1.0	0.0	0.0	0.0	0.0	0.0	No	M
1	1.0	1.0	0.0	0.0	0.0	0.0	No	M
2	0.0	1.0	0.0	0.0	0.0	0.0	Yes	M
3	1.0	1.0	0.0	0.0	0.0	1.0	Yes	M
4	1.0	0.0	0.0	0.0	0.0	0.0	Yes	Fem
...	...	...	...	...	...	...	...	
211424	0.0	1.0	0.0	0.0	0.0	1.0	Yes	M
211425	0.0	0.0	0.0	0.0	0.0	1.0	No	M
211426	0.0	0.0	0.0	0.0	0.0	1.0	No	I
211427	0.0	0.0	0.0	0.0	0.0	1.0	No	Fem
211428	0.0	0.0	0.0	0.0	0.0	1.0	No	Fem

211429 rows × 9 columns

In [29]:

```
df.head(20)
```

Out[29]:

	cough	fever	sore_throat	shortness_of_breath	head_ache	corona_result	age_60_and_above	gender
0	1.0	0.0	0.0	0.0	0.0	0.0	No	Male
1	1.0	1.0	0.0	0.0	0.0	0.0	No	Male
2	0.0	1.0	0.0	0.0	0.0	0.0	Yes	Male
3	1.0	1.0	0.0	0.0	0.0	1.0	Yes	Male
4	1.0	0.0	0.0	0.0	0.0	0.0	Yes	Female
5	1.0	1.0	0.0	0.0	0.0	1.0	Yes	Male
6	0.0	0.0	0.0	0.0	0.0	0.0	No	Male
7	1.0	1.0	1.0	0.0	1.0	0.0	No	Female
8	1.0	0.0	1.0	0.0	0.0	0.0	No	Female
9	1.0	1.0	1.0	1.0	0.0	0.0	No	Female
10	1.0	0.0	0.0	0.0	0.0	0.0	No	Male
11	1.0	0.0	0.0	0.0	0.0	0.0	No	Female
12	1.0	0.0	0.0	0.0	0.0	0.0	No	Female
13	0.0	0.0	0.0	0.0	0.0	0.0	No	Male
14	0.0	0.0	0.0	0.0	0.0	0.0	No	Female
15	0.0	0.0	0.0	0.0	0.0	0.0	No	Male
16	1.0	1.0	0.0	1.0	0.0	0.0	No	Male
17	0.0	1.0	0.0	0.0	0.0	1.0	No	Female
18	0.0	1.0	0.0	0.0	0.0	1.0	No	Female
19	0.0	0.0	0.0	0.0	0.0	0.0	No	Female

Data Preprocessing

Removing null values

In [30]:

```
df.isnull().sum()
```

Out[30]:

```
cough                   243
fever                   243
sore_throat             243
shortness_of_breath     243
head_ache               243
corona_result           243
age_60_and_above        243
gender                 5371
test_indication         243
dtype: int64
```

In [31]:

```
df=df.dropna()
```

In [32]:

```
df.isnull().sum()
```

Out[32]:

```
cough                  0
fever                  0
sore_throat            0
shortness_of_breath    0
head_ache              0
corona_result          0
age_60_and_above       0
gender                 0
test_indication        0
dtype: int64
```

There are no null values

In [33]:

```
np.shape(df)
```

Out[33]:

```
(206058, 9)
```

Replacing categorical variables with numbers

In [34]:

```
df=df.replace(to_replace="No", value=0.0)
df=df.replace(to_replace="Yes", value=1.0)
```

In [35]:

```
df=df.replace(to_replace="Male", value=0.0)
df=df.replace(to_replace="Female",value=1.0)
```

In [36]:

```
df=df.replace(to_replace="Other", value=1.0)
df=df.replace(to_replace="Abroad", value=2.0)
df=df.replace(to_replace="Contact with confirmed",value=3.0)
```

In [37]:

```
df.head(10)
```

Out[37]:

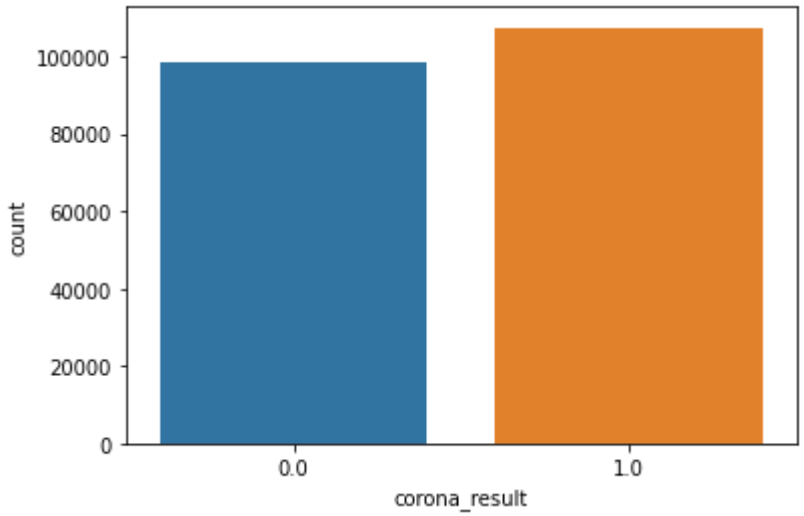
	cough	fever	sore_throat	shortness_of_breath	head_ache	corona_result	age_60_and_above	gender
0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0
3	1.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0
4	1.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0
5	1.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0
6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7	1.0	1.0	1.0	0.0	1.0	0.0	0.0	1.0
8	1.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0
9	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0

Distribution of positive and negative results

In [38]:

```
sn.countplot(df["corona_result"])
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWa
rning: Pass the following variable as a keyword arg: x. From version 0.12,
the only valid positional argument will be `data`, and passing other argum
ents without an explicit keyword will result in an error or misinterpretat
ion.
  FutureWarning
```



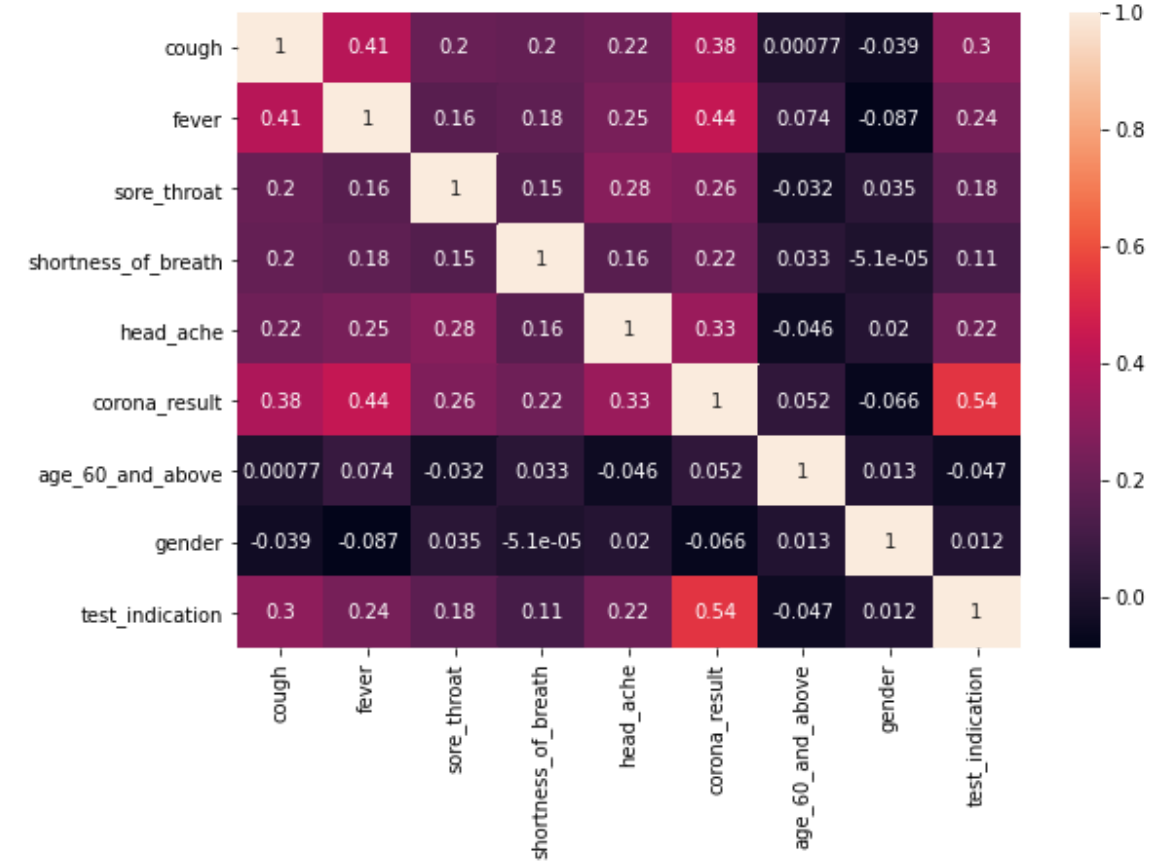
Checking for relationship among variables

In [39]:

```
plt.figure(figsize=(9,6))
c=df.corr()
sn.heatmap(c,xticklabels=c.columns,yticklabels=c.columns,annot=True)
c
```

Out[39]:

	cough	fever	sore_throat	shortness_of_breath	head_ache	corona_result	age
cough	1.000000	0.405062	0.202896	0.195795	0.220778	0.377659	
fever	0.405062	1.000000	0.160196	0.180182	0.247458	0.437327	
sore_throat	0.202896	0.160196	1.000000	0.150116	0.282549	0.264471	
shortness_of_breath	0.195795	0.180182	0.150116	1.000000	0.162358	0.222676	
head_ache	0.220778	0.247458	0.282549	0.162358	1.000000	0.330164	
corona_result	0.377659	0.437327	0.264471	0.222676	0.330164	1.000000	
age_60_and_above	0.000766	0.073613	-0.032025	0.032874	-0.045978	0.051769	
gender	-0.039114	-0.087300	0.035454	-0.000051	0.019952	-0.066191	
test_indication	0.301624	0.241511	0.175073	0.112905	0.217364	0.540474	



Train-Test split

In [40]:

```
df.columns
```

Out[40]:

```
Index(['cough', 'fever', 'sore_throat', 'shortness_of_breath', 'head_ach
e',
       'corona_result', 'age_60_and_above', 'gender', 'test_indication'],
      dtype='object')
```

In [41]:

```
X=df[['cough', 'fever', 'sore_throat', 'shortness_of_breath', 'head_ache',
      'age_60_and_above', 'gender', 'test_indication']]
```

In [42]:

```
y=df['corona_result']
```

In [43]:

```
X_train, X_test, y_train, y_test = train_test_split(X,y)
```

Naive Bayes

In [58]:

```
naives_model = GaussianNB()
naives_model.fit(X_train,y_train)
```

Out[58]:

```
GaussianNB(priors=None, var_smoothing=1e-09)
```

In [59]:

```
Y_hat = naives_model.predict(X_test)
Y_hat
```

Out[59]:

```
array([1., 1., 0., ..., 0., 0., 0.])
```

In [60]:

```
model_acc = accuracy_score(y_test, Y_hat)
print('The accuracy of our naive model is: %0.2f'% model_acc)
```

```
The accuracy of our naive model is: 0.83
```

Decision Tree

In [63]:

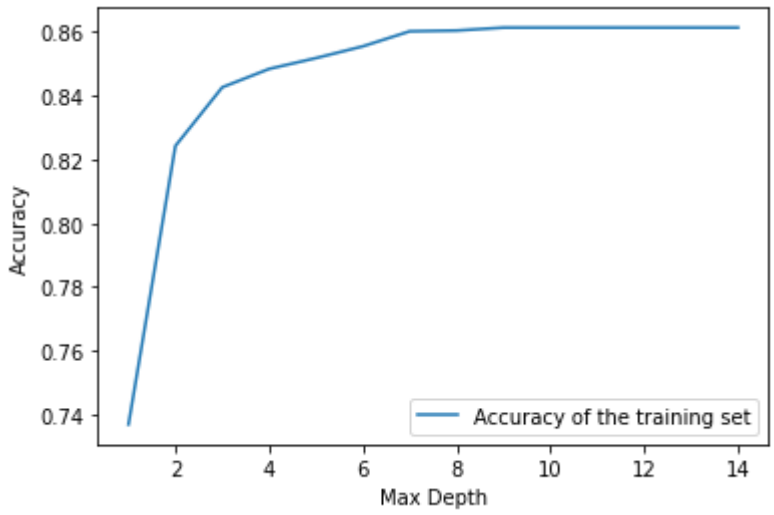
```
training_accuracy = []
test_accuracy = []
max_dep = range(1,15)

for md in max_dep:
    tree = DecisionTreeClassifier(max_depth=md,random_state=0)
    tree.fit(X_train,y_train)
    training_accuracy.append(tree.score(X_train, y_train))
    test_accuracy.append(tree.score(X_test, y_test))

plt.plot(max_dep,training_accuracy, label='Accuracy of the training set')
plt.ylabel('Accuracy')
plt.xlabel('Max Depth')
plt.legend()
```

Out[63]:

```
<matplotlib.legend.Legend at 0x7fca4560bb90>
```



In [70]:

```
tree = DecisionTreeClassifier(max_depth=3,random_state=0)
tree.fit(X_train,y_train)
```

Out[70]:

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=3, max_features=None, max_leaf_nodes=Non
e,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=0, splitter='best')
```

In [71]:

```
tree.score(X_test, y_test)
```

Out[71]:

```
0.8432301271474328
```

Gradient Boosting

In [44]:

```
from sklearn.ensemble import GradientBoostingClassifier
clf=GradientBoostingClassifier()
```

In [45]:

```
clf.fit(X_train,y_train)
```

Out[45]:

```
GradientBoostingClassifier(ccp_alpha=0.0, criterion='friedman_mse', init=N
one,
                           learning_rate=0.1, loss='deviance', max_depth=
3,
                           max_features=None, max_leaf_nodes=None,
                           min_impurity_decrease=0.0, min_impurity_split=N
one,
                           min_samples_leaf=1, min_samples_split=2,
                           min_weight_fraction_leaf=0.0, n_estimators=100,
                           n_iter_no_change=None, presort='deprecated',
                           random_state=None, subsample=1.0, tol=0.0001,
                           validation_fraction=0.1, verbose=0,
                           warm_start=False)
```

In [46]:

```
pred_gb=clf.predict(X_test)
```

In [47]:

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,pred_gb)
```

Out[47]:

```
0.8617878287877317
```

In [48]:

```
confusion_matrix(y_test,pred_gb)
```

Out[48]:

```
array([[22342,  2202],
       [ 4918, 22053]])
```

In [49]:

```
print(classification_report(y_test,pred_gb))
```

```
              precision    recall  f1-score   support

         0.0       0.82      0.91      0.86     24544
         1.0       0.91      0.82      0.86     26971

    accuracy                           0.86     51515
   macro avg       0.86      0.86      0.86     51515
weighted avg       0.87      0.86      0.86     51515
```

RandomForest Classifier

In [50]:

```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier(max_depth=2)
```

In [51]:

```
rfc.fit(X_train,y_train)
```

Out[51]:

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=2, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

In [52]:

```
pred_rfc=rfc.predict(X_test)
```

In [53]:

```
accuracy_score(y_test,pred_rfc)
```

Out[53]:

```
0.8467630787149374
```

In [54]:

```
confusion_matrix(y_test,pred_rfc)
```

Out[54]:

```
array([[21602,  2942],
       [ 4952, 22019]])
```

In [55]:

```
print(classification_report(y_test,pred_rfc))
```

```
              precision    recall  f1-score   support

         0.0       0.81      0.88      0.85     24544
         1.0       0.88      0.82      0.85     26971

    accuracy                           0.85     51515
   macro avg       0.85      0.85      0.85     51515
weighted avg       0.85      0.85      0.85     51515
```

Accuracy Results

- **Naive Bayes- 83%**
- **Decision Tree- 84.3%**
- **Gradient Boosting- 86.27%**
- **RandomForest Classifier- 84.58%**

Gradient boosting gives the best results