DEVELOPMENT OF MACHINE LEARNING FRAMEWORK TO IDENTIFY ACUTE EXACEBERATION CHRONIC OBSTRUCTIVE PULMONARY DISEASE

importing libraries

In [214…

```
import pandas as pd          # for importing and managing dataset
import numpy as np           #  for mathematical operaation


from sklearn.impute import SimpleImputer     #to remove null values

from sklearn.model_selection import train_test_split  # to train and test the dataset

from sklearn.preprocessing import StandardScaler     #feature scaling using standardization

from sklearn.linear_model import LogisticRegression  # logistic regression
from sklearn.ensemble import RandomForestClassifier  # random forest
from sklearn.neighbors import KNeighborsClassifier  # k neighbourclassifier
from sklearn.tree import DecisionTreeClassifier     #decision tree
from sklearn.naive_bayes import GaussianNB          #naive bayes
from sklearn.svm import SVC                         #support vector machine

from sklearn.metrics import accuracy_score          #  accuracy score


from sklearn import metrics

import seaborn as sns
import matplotlib.pyplot as plt
```

importing data set

In [215…

```
df=pd.read_excel('health_records.xlsx')
df
```

Out[215…

	participant	date	peakflow	wheese	breathlessness	cough	sputum	antibiotics	steroids	oxygen	inhalers	home	gp	hospital	sleep
0	1	2015-05-08	300.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	1	2015-05-09	350.0	1.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	1	2015-05-10	325.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3	1	2015-05-11	325.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	1	2015-05-12	350.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
17079	130	2017-10-07	140.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
17080	130	2017-10-08	150.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
17081	130	2017-09-10	110.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
17082	130	2017-10-10	130.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
17083	130	2017-11-10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

17084 rows × 18 columns

In [216…

```
df.head()
```

Out[216…

	participant	date	peakflow	wheese	breathlessness	cough	sputum	antibiotics	steroids	oxygen	inhalers	home	gp	hospital	sleep_dist
0	1	2015-05-08	300.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

1	1	2015-05-09	350.0	1.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
2	1	2015-05-10	325.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	1	2015-05-11	325.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
4	1	2015-05-12	350.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

In [217…
```
df.tail()
```

Out[217…

	participant	date	peakflow	wheese	breathlessness	cough	sputum	antibiotics	steroids	oxygen	inhalers	home	gp	hospital	sleep
17079	130	2017-10-07	140.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
17080	130	2017-10-08	150.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
17081	130	2017-09-10	110.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
17082	130	2017-10-10	130.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
17083	130	2017-11-10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

In [218…
```
df.shape
```

Out[218…
```
(17084, 18)
```

In [219…
```
df.describe()
```

Out[219…

	participant	peakflow	wheese	breathlessness	cough	sputum	antibiotics	steroids	oxygen	
count	17084.000000	16342.000000	16566.000000	16561.000000	16566.000000	16564.000000	16578.000000	16578.000000	16576.000000	16586
mean	64.007141	255.337474	0.173126	0.378298	0.323494	0.163125	0.056098	0.038123	0.003197	0
std	36.816948	131.978989	0.521524	0.802734	0.755757	0.530039	0.242624	0.199521	0.073207	0
min	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0
25%	32.000000	160.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0
50%	62.000000	230.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0
75%	93.000000	340.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0
max	130.000000	5100.000000	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000	1

In [220…
```
df.dtypes
```

Out[220…
```
participant              int64
date            datetime64[ns]
peakflow               float64
wheese                 float64
breathlessness         float64
cough                  float64
sputum                 float64
antibiotics            float64
steroids               float64
oxygen                 float64
inhalers               float64
home                   float64
gp                     float64
hospital               float64
sleep_dist             float64
leavehouse             float64
takemonitor            float64
exacerbation           float64
dtype: object
```

In [221…
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17084 entries, 0 to 17083
Data columns (total 18 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   participant     17084 non-null  int64
 1   date            17084 non-null  datetime64[ns]
 2   peakflow        16342 non-null  float64
 3   wheese          16566 non-null  float64
 4   breathlessness  16561 non-null  float64
 5   cough           16566 non-null  float64
 6   sputum          16564 non-null  float64
 7   antibiotics     16578 non-null  float64
 8   steroids        16578 non-null  float64
 9   oxygen          16576 non-null  float64
 10  inhalers        16586 non-null  float64
 11  home            16587 non-null  float64
 12  gp              16587 non-null  float64
 13  hospital        16587 non-null  float64
 14  sleep_dist      16570 non-null  float64
 15  leavehouse      16199 non-null  float64
 16  takemonitor     16201 non-null  float64
 17  exacerbation    16409 non-null  float64
dtypes: datetime64[ns](1), float64(16), int64(1)
memory usage: 2.3 MB
```

dropping date column

In [222…

```
newdf=df.drop('date',axis='columns')
newdf
```

Out[222…

	participant	peakflow	wheese	breathlessness	cough	sputum	antibiotics	steroids	oxygen	inhalers	home	gp	hospital	sleep_dist
0	1	300.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	1	350.0	1.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
2	1	325.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	1	325.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
4	1	350.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
17079	130	140.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
17080	130	150.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
17081	130	110.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
17082	130	130.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
17083	130	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

17084 rows × 17 columns

In [223…

```
newdf.dtypes
```

Out[223…

```
participant         int64
peakflow          float64
wheese            float64
breathlessness    float64
cough             float64
sputum            float64
antibiotics       float64
steroids          float64
oxygen            float64
inhalers          float64
home              float64
gp                float64
hospital          float64
sleep_dist        float64
leavehouse        float64
takemonitor       float64
exacerbation      float64
dtype: object
```

handling missing data

handling missing data

In [224…
```
newdf.isnull().sum()
```

Out[224…
```
participant         0
peakflow          742
wheese            518
breathlessness    523
cough             518
sputum            520
antibiotics       506
steroids          506
oxygen            508
inhalers          498
home              497
gp                497
hospital          497
sleep_dist        514
leavehouse        885
takemonitor       883
exacerbation      675
dtype: int64
```

In [225…
```
newdf.isnull().sum().sum()
```

Out[225…
```
9287
```

filling null values with mean

In [226…
```
imputer=SimpleImputer(missing_values=np.nan,strategy='mean')
imputer.fit(newdf)
newdf=imputer.transform(newdf)
```

In [227…
```
newdf_1=pd.DataFrame(newdf)
newdf_1
```

Out[227…

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	1.0	300.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.697204	0.384668	0.0
1	1.0	350.0	1.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.697204	0.384668	0.0
2	1.0	325.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.697204	0.384668	0.0
3	1.0	325.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.697204	0.384668	0.0
4	1.0	350.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.697204	0.384668	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
17079	130.0	140.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.000000	0.000000	0.0
17080	130.0	150.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.000000	0.000000	0.0
17081	130.0	110.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.000000	0.000000	0.0
17082	130.0	130.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.000000	1.000000	0.0
17083	130.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.000000	1.000000	0.0

17084 rows × 17 columns

In [228…
```
newdf_1.isnull().sum().sum()
```

Out[228…
```
0
```

In [229…
```
newdf_1[16]= newdf_1[16].apply(np.int64)
newdf_1.dtypes
```

Out[229…
```
0     float64
1     float64
2     float64
3     float64
```

```
4     float64
5     float64
6     float64
7     float64
8     float64
9     float64
10    float64
11    float64
12    float64
13    float64
14    float64
15    float64
16      int64
dtype: object
```

extracting independent and dependent attributes

In [230…
```
x=newdf_1.iloc[:,:16]
y=newdf_1.iloc[:,16]
```

In [231…
```
x
```

Out[231…

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1.0	300.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.697204	0.384668
1	1.0	350.0	1.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.697204	0.384668
2	1.0	325.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.697204	0.384668
3	1.0	325.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.697204	0.384668
4	1.0	350.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.697204	0.384668
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
17079	130.0	140.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.000000	0.000000
17080	130.0	150.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.000000	0.000000
17081	130.0	110.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.000000	0.000000
17082	130.0	130.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.000000	1.000000
17083	130.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.000000	1.000000

17084 rows × 16 columns

In [232…
```
y
```

Out[232…
```
0        0
1        0
2        0
3        0
4        0
       ..
17079    0
17080    0
17081    0
17082    0
17083    0
Name: 16, Length: 17084, dtype: int64
```

splitting data into trainig and testing

In [233…
```
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=3)
```

In [234…
```
print(x_train)
print(x_test)
print(y_train)
print(y_test)
```

```
          0      1    2    3    4    5    6    7    8    9    10   11   12  \
```

```
1110     9.0  160.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
1453    12.0  150.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
4261    32.0  200.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
10733   79.0  250.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
6214    47.0  100.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
...      ...    ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...
6400    49.0  180.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
15288  117.0  130.0  0.0  0.0  2.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
11513   84.0   60.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
1688    15.0  180.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
5994    45.0  220.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0

        13   14   15
1110   0.0  0.0  0.0
1453   0.0  0.0  0.0
4261   0.0  0.0  0.0
10733  0.0  1.0  1.0
6214   0.0  0.0  0.0
...    ...  ...  ...
6400   0.0  1.0  0.0
15288  1.0  1.0  1.0
11513  0.0  0.0  0.0
1688   0.0  0.0  0.0
5994   0.0  1.0  0.0

[13667 rows x 16 columns]
          0           1         2         3         4         5         6   \
16455  126.0  150.000000  0.000000  0.000000  0.000000  0.000000  0.000000
9654    69.0  360.000000  0.000000  0.000000  0.000000  0.000000  0.000000
5511    42.0  350.000000  1.000000  0.000000  1.000000  0.000000  0.000000
14705  111.0  150.000000  0.000000  3.000000  0.000000  0.000000  0.000000
15653  118.0  120.000000  0.000000  0.000000  0.000000  0.000000  0.000000
...      ...         ...       ...       ...       ...       ...       ...
11285   83.0  460.000000  0.000000  0.000000  3.000000  0.000000  1.000000
4619    35.0  350.000000  0.000000  0.000000  0.000000  0.000000  0.000000
12731   93.0  255.337474  0.173126  0.378298  0.323494  0.163125  0.056098
622      6.0  230.000000  0.000000  0.000000  0.000000  0.000000  0.000000
2246    19.0  310.000000  0.000000  0.000000  0.000000  0.000000  0.000000

             7         8         9         10        11        12        13  \
16455  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
9654   0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  1.000000
5511   0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
14705  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
15653  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
...         ...       ...       ...       ...       ...       ...       ...
11285  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
4619   0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
12731  0.038123  0.003197  0.019414  0.013143  0.032435  0.006632  0.119855
622    0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
2246   0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000

             14        15
16455  1.000000  1.000000
9654   1.000000  0.000000
5511   1.000000  0.000000
14705  1.000000  0.000000
15653  1.000000  1.000000
...         ...       ...
11285  1.000000  0.000000
4619   0.000000  0.000000
12731  0.697204  0.384668
622    0.000000  0.000000
2246   1.000000  1.000000

[3417 rows x 16 columns]
1110     0
1453     0
4261     0
10733    0
6214     0
        ..
6400     0
15288    1
11513    0
1688     0
5994     0
Name: 16, Length: 13667, dtype: int64
16455    0
9654     0
5511     0
14705    1
15653    0
        ..
11285    1
4619     0
12731    0
622      0
2246     0
```

```
Name: 16, Length: 3417, dtype: int64
```

feature scaling by using standardization

In [235…
```
from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
x_test= st_x.transform(x_test)
```

applying models

1)logisitc regression

In [236…
```
model_lr = LogisticRegression()
model_lr.fit(x_train, y_train)
```

Out[236…
```
LogisticRegression()
```

accuracy score

In [237…
```
y_pred= model_lr.predict(x_test)
from sklearn.metrics import accuracy_score
lr=accuracy_score(y_test,y_pred)*100
print("accuracy score logistic regression:",lr)
```

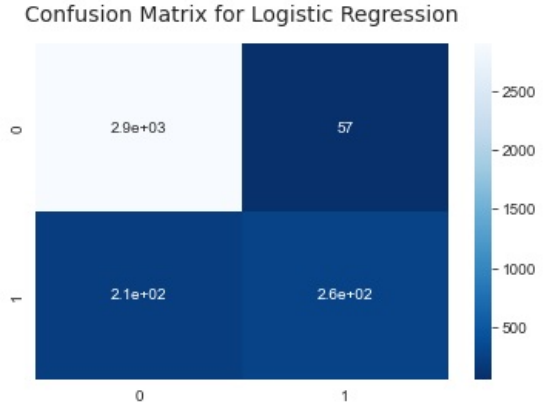
```
accuracy score logistic regression: 92.2739244951712
```

confusion matrix

In [238…
```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,y_pred)
```

Out[238…
```
array([[2894,   57],
       [ 207,  259]], dtype=int64)
```

In [239…
```
from sklearn.metrics import confusion_matrix
sns.set_style("white")
y_pred = model_lr.predict(x_test)
cf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(cf_matrix, annot=True, cmap="Blues_r")
plt.title("Confusion Matrix for Logistic Regression", fontsize=14, fontname="Helvetica", y=1.04);
```



2)random forest

In [240…
```
model_rf =RandomForestClassifier(n_estimators=100,random_state=0)
model_rf.fit(x_train, y_train)
```

Out[240…
```
RandomForestClassifier(random_state=0)
```

accuracy score

In [241…
```
y_pred= model_rf.predict(x_test)
from sklearn.metrics import accuracy_score
rf=accuracy_score(y_test,y_pred)*100
print("accuracy score random forest:",rf)
```

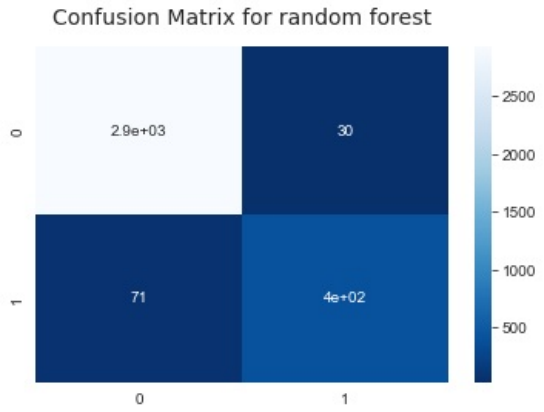
```
accuracy score random forest: 97.04419081065262
```

confusion matrix

In [242…
```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,y_pred)
```

Out[242…
```
array([[2921,   30],
       [  71,  395]], dtype=int64)
```

In [243…
```
from sklearn.metrics import confusion_matrix
sns.set_style("white")
y_pred = model_rf.predict(x_test)
cf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(cf_matrix, annot=True, cmap="Blues_r")
plt.title("Confusion Matrix for random forest", fontsize=14, fontname="Helvetica", y=1.04);
```



3)k nearest neighbours

In [244…
```
model_knn=KNeighborsClassifier(n_neighbors=5, metric='minkowski', p=2 )
model_knn.fit(x_train, y_train)
```

Out[244…
```
KNeighborsClassifier()
```

accuracy score

In [245…]
```
y_pred= model_knn.predict(x_test)
from sklearn.metrics import accuracy_score
knn=accuracy_score(y_test,y_pred)*100
print("accuracy score k nearest neighbours:",knn)
```

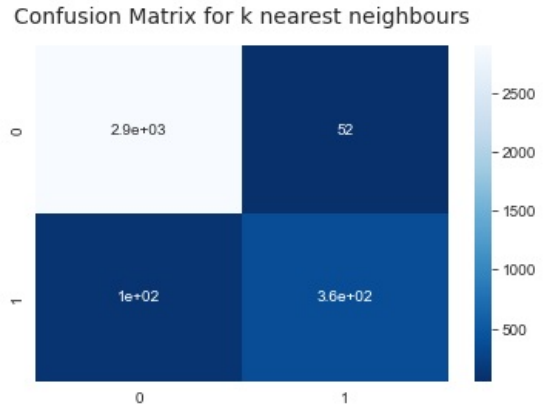
```
accuracy score k nearest neighbours: 95.52238805970148
```

confusion matrix

In [246…]
```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,y_pred)
```

Out[246…]
```
array([[2899,   52],
       [ 101,  365]], dtype=int64)
```

In [247…]
```
from sklearn.metrics import confusion_matrix
sns.set_style("white")
y_pred = model_knn.predict(x_test)
cf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(cf_matrix, annot=True, cmap="Blues_r")
plt.title("Confusion Matrix for k nearest neighbours", fontsize=14, fontname="Helvetica", y=1.04);
```



4)decision tree

In [248…]
```
model_dt= DecisionTreeClassifier(criterion='entropy', random_state=0)
model_dt.fit(x_train, y_train)
```

Out[248…]
```
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

accuracy score

In [249…]
```
y_pred= model_dt.predict(x_test)
from sklearn.metrics import accuracy_score
dt=accuracy_score(y_test,y_pred)*100
print("accuracy score decision tree:",dt)
```

```
accuracy score decision tree: 96.7515364354697
```

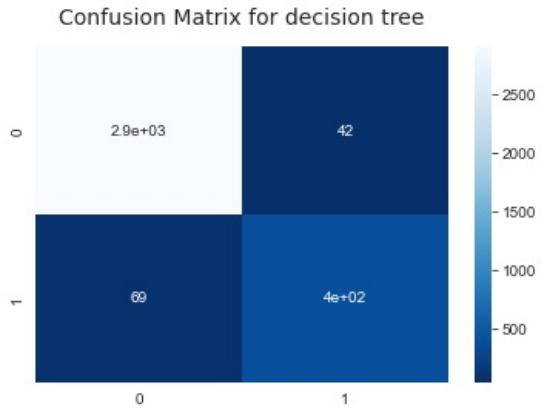
confusion matrix

In [251…]
```
import seaborn as sns
```

```
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,y_pred)
```

Out[251…
```
array([[2909,   42],
       [  69,  397]], dtype=int64)
```

In [252…
```
from sklearn.metrics import confusion_matrix
sns.set_style("white")
y_pred = model_dt.predict(x_test)
cf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(cf_matrix, annot=True, cmap="Blues_r")
plt.title("Confusion Matrix for decision tree", fontsize=14, fontname="Helvetica", y=1.04);
```



5)naive bayes

In [253…
```
model_nb = GaussianNB()
model_nb.fit(x_train, y_train)
```

Out[253…
```
GaussianNB()
```

accuracy score

In [254…
```
y_pred= model_nb.predict(x_test)
from sklearn.metrics import accuracy_score
nb=accuracy_score(y_test,y_pred)*100
print("accuracy score naive bayes:",nb)
```

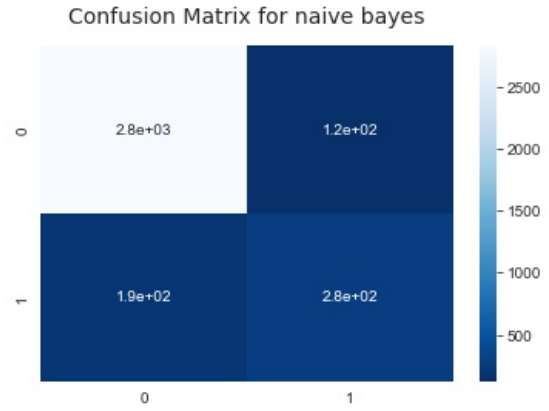
```
accuracy score naive bayes: 90.83991805677495
```

confusion matrix

In [255…
```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,y_pred)
```

Out[255…
```
array([[2827,  124],
       [ 189,  277]], dtype=int64)
```

In [256…
```
from sklearn.metrics import confusion_matrix
sns.set_style("white")
y_pred = model_nb.predict(x_test)
cf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(cf_matrix, annot=True, cmap="Blues_r")
plt.title("Confusion Matrix for naive bayes", fontsize=14, fontname="Helvetica", y=1.04);
```



6)support vector machine

In [257…
```
model_svm = SVC(kernel='linear', random_state=5)
model_svm.fit(x_train, y_train)
```

Out[257…
```
SVC(kernel='linear', random_state=5)
```

accuracy score

In [258…
```
y_pred= model_svm.predict(x_test)
from sklearn.metrics import accuracy_score
svm=accuracy_score(y_test,y_pred)*100
print("accuracy score support vector machine:",svm)
```

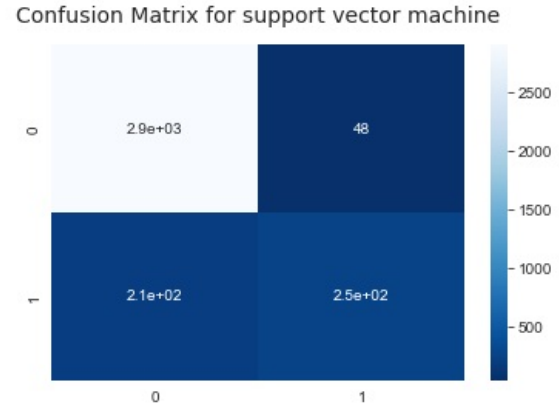
```
accuracy score support vector machine: 92.33245537020778
```

confusion matrix

In [259…
```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,y_pred)
```

Out[259…
```
array([[2903,   48],
       [ 214,  252]], dtype=int64)
```

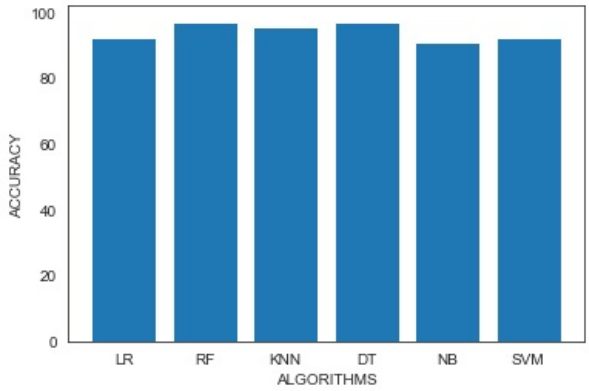
In [260…
```
from sklearn.metrics import confusion_matrix
sns.set_style("white")
y_pred = model_svm.predict(x_test)
cf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(cf_matrix, annot=True, cmap="Blues_r")
plt.title("Confusion Matrix for support vector machine", fontsize=14, fontname="Helvetica", y=1.04);
```



comparision of algorithms

In [261…

```
plt.bar(['LR','RF','KNN','DT','NB','SVM'],[lr,rf,knn,dt,nb,svm])
plt.xlabel("ALGORITHMS")
plt.ylabel("ACCURACY")
plt.show()
```



using random forest to predict

instance example

In [262…

```
input_data=[]
for i in range(16):
    if(i==0):
        print("PARTICIPANT :")
        ele=float(input())
        input_data.append(ele)
    elif(i==1):
        print("PEAKFLOW :")
        ele=float(input())
        input_data.append(ele)
    elif(i==2):
        print("WHEESE :")
        ele=float(input())
        input_data.append(ele)
    elif(i==3):
        print("BREATHLESSNESS :")
        ele=float(input())
        input_data.append(ele)
    elif(i==4):
        print("COUGH:")
        ele=float(input())
        input_data.append(ele)
    elif(i==5):
        print("SPUTUM :")
        ele=float(input())
        input_data.append(ele)
    elif(i==6):
        print("ANTIBIOTICS :")
        ele=float(input())
        input_data.append(ele)
    elif(i==7):
        print("STEROIDS :")
        ele=float(input())
        input_data.append(ele)
    elif(i==8):
        print("OXYGEN :")
        ele=float(input())
        input_data.append(ele)
    elif(i==9):
        print("INHALERS :")
        ele=float(input())
        input_data.append(ele)
    elif(i==10):
        print("HOME :")
        ele=float(input())
        input_data.append(ele)
    elif(i==11):
        print("GP :")
        ele=float(input())
        input_data.append(ele)
```

```
    elif(i==12):
        print("HOSPITAL:")
        ele=float(input())
        input_data.append(ele)
    elif(i==13):
        print("sleep_dist:")
        ele=float(input())
        input_data.append(ele)
    elif(i==14):
        print("leavehouse:")
        ele=float(input())
        input_data.append(ele)
    elif(i==15):
        print("takemonitor:")
        ele=float(input())
        input_data.append(ele)



input_data=tuple(input_data) #LIST TO TUPLE CONVERSION.....
print("-------------------------------------------------------------------------------------------------------------------")
print("THE INPUT_DATA IS ------------------>",input_data)
```

```
PARTICIPANT :
22
PEAKFLOW :
135
WHEESE :
3
BREATHLESSNESS :
3
COUGH:
1
SPUTUM :
3
ANTIBIOTICS :
1
STEROIDS :
0
OXYGEN :
0
INHALERS :
0
HOME :
0
GP :
0
HOSPITAL:
0
sleep_dist:
0
leavehouse:
1
takemonitor:
1
-------------------------------------------------------------------------------------------------------------------
-----
THE INPUT_DATA IS ------------------> (22.0, 135.0, 3.0, 3.0, 1.0, 3.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1
.0, 1.0)
```

In [263…

```
input_data_as_numpy_array= np.asarray(input_data)

# reshape the numpy array as we are predicting for only on instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = model_rf.predict(input_data_reshaped)
print(prediction)

if (prediction[0]== 0):
  print('The Person does not have a AECOPD')
else:
  print('The Person has AECOPD')
```

```
[1]
The Person has AECOPD
```

In [ ]: