Lung Cancer Detection

INTRODUCTION TO DATA SCIENCE(BTIBM505)

PROJECT:LUNG CANCER DETECTION

MADE BY:

GAUTAM JAISWAL(20100BTCSAII07160)

OJAS KHATAVKAR (20100BTCSAII07178)

SOPHIYA SALEEMA QURESHI (20100BTCSAII07190)

KAMAKSHI SHARMA (20100BTCSAII07170)

SEC-D (AI) 3RD YEAR 5TH SEM

SESSION 2022-23

About Dataset The effectiveness of the cancer prediction system helps people to know their cancer risk wi a low cost and it also helps the people to take the appropriate decision based on their cancer risk status. The data is collected from the website online lung cancer prediction system.

In [35]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [36]:
```
df1=pd.read_csv("lung_cancer.csv")
df1.columns
```

Out[36]:
```
Index(['GENDER', 'AGE', 'SMOKING', 'YELLOW_FINGERS', 'ANXIETY',
       'PEER_PRESSURE', 'CHRONIC DISEASE', 'FATIGUE ', 'ALLERGY ', 'WHEEZING',
       'ALCOHOL CONSUMING', 'COUGHING', 'SHORTNESS OF BREATH',
       'SWALLOWING DIFFICULTY', 'CHEST PAIN', 'LUNG_CANCER'],
      dtype='object')
```

In [37]:
```
print("HERE BEFORE LABEL ENCODING")
print("2 denotes 'YES, person have cancer'\n'1 denotes NO,person does not have canc
df1.head()
```

```
HERE BEFORE LABEL ENCODING
2 denotes 'YES, person have cancer'
'1 denotes NO,person does not have cancer
```

Out[37]:

	GENDER	AGE	SMOKING	YELLOW_FINGERS	ANXIETY	PEER_PRESSURE	CHRONIC DISEASE	FATIGUE
0	M	69	1	2	2	1	1	2
1	M	74	2	1	1	1	2	2
2	F	59	1	1	1	2	1	2
3	M	63	2	2	2	1	1	1
4	F	63	1	2	1	1	1	1

In [38]:
```
df1.describe()
```

Out[38]:

	AGE	SMOKING	YELLOW_FINGERS	ANXIETY	PEER_PRESSURE	CHRONIC DISEASE	FATI
count	309.000000	309.000000	309.000000	309.000000	309.000000	309.000000	309.00
mean	62.673139	1.563107	1.569579	1.498382	1.501618	1.504854	1.67
std	8.210301	0.496806	0.495938	0.500808	0.500808	0.500787	0.46
min	21.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.00
25%	57.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.00
50%	62.000000	2.000000	2.000000	1.000000	2.000000	2.000000	2.00
75%	69.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.00
max	87.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.00

In [39]:
```
df1["LUNG_CANCER"].unique()
```

Out[39]:
```
array(['YES', 'NO'], dtype=object)
```

In [40]:
```
from sklearn.preprocessing import LabelEncoder
lb=LabelEncoder()
df1['LUNG_CANCER']=lb.fit_transform (df1[ 'LUNG_CANCER'])
df1['GENDER']=lb.fit_transform (df1[ 'GENDER'])
x=df1[[ 'GENDER', 'AGE', 'SMOKING', 'YELLOW_FINGERS', 'ANXIETY','PEER_PRESSURE', 'C
       'ALCOHOL CONSUMING', 'COUGHING', 'SHORTNESS OF BREATH',
       'SWALLOWING DIFFICULTY', 'CHEST PAIN']]
y=df1[ 'LUNG_CANCER']
from sklearn.model_selection import train_test_split
x_train,x_test,y_train, y_test=train_test_split(x,y,test_size=.25)
```

In [41]:
```
print("HERE AFTER LABEL ENCODING")
print("1 denotes 'YES, person have cancer'\n'0 denotes NO,person does not have canc
df1.head()
```

```
HERE AFTER LABEL ENCODING
1 denotes 'YES, person have cancer'
'0 denotes NO,person does not have cancer
```

Out[41]:

	GENDER	AGE	SMOKING	YELLOW_FINGERS	ANXIETY	PEER_PRESSURE	CHRONIC DISEASE	FATIGUE
0	1	69	1	2	2	1	1	2
1	1	74	2	1	1	1	2	2
2	0	59	1	1	1	2	1	2
3	1	63	2	2	2	1	1	1
4	0	63	1	2	1	1	1	1

linear Regression

In [42]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train, y_test=train_test_split(x,y,test_size=.25)
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train, y_train)
y_pred=lr.predict(x_test)
print("1 denotes 'YES, person have cancer'\n'0 denotes NO,person does not have canc
print (f'first 5 actual_y: {y_test.values[:5]} predicted y: {y_pred[:5]}')

print('Model_score :',lr.score (x_test,y_test))
linearReg_accuracy=lr.score(x_test,y_test)
#sns.heatmap(a, annot=True)
#plt.show()
```

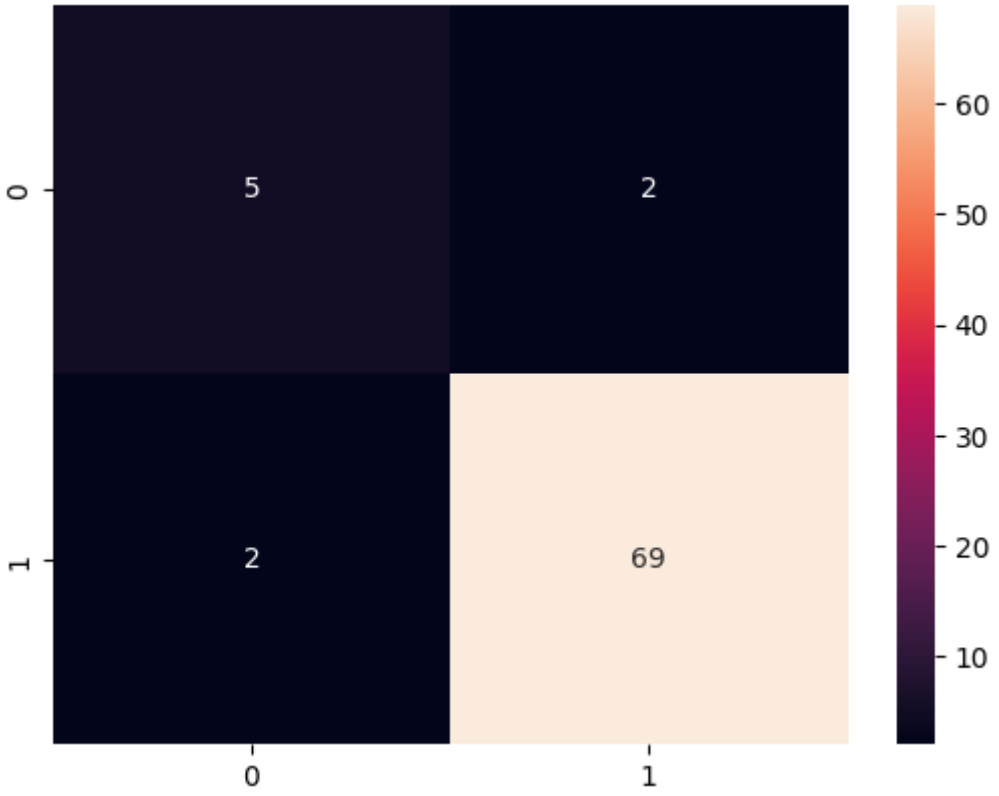
```
1 denotes 'YES, person have cancer'
'0 denotes NO,person does not have cancer
first 5 actual_y: [1 1 1 1 1] predicted y: [0.90192815 1.19227659 1.26731771 0.985
90128 0.99406915]
Model_score : 0.2079084725452871
```

logistic regression

In [43]:
```
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
lr.fit(x_train, y_train)
y_pred=lr.predict(x_test)
print("1 denotes 'YES, person have cancer'\n'0 denotes NO,person does not have canc
print (f'first 5 actual_y: {y_test.values[:5]} predicted y: {y_pred[:5]}')
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
print('confusion_matrix')
a=confusion_matrix (y_test,y_pred)
print(a)
print('classification_report')
print(classification_report (y_test,y_pred))
print('accuracy_score-', accuracy_score (y_test,y_pred))
print('Model_score -',lr.score (x_test,y_test))
logisticReg_accuracy=accuracy_score(y_test,y_pred)
sns.heatmap(a, annot=True)
plt.show()
```

```
1 denotes 'YES, person have cancer'
'0 denotes NO,person does not have cancer
first 5 actual_y: [1 1 1 1 1] predicted y: [1 1 1 1 1]
confusion_matrix
[[ 5  2]
 [ 2 69]]
classification_report
              precision    recall  f1-score   support

           0       0.71      0.71      0.71         7
           1       0.97      0.97      0.97        71

    accuracy                           0.95        78
   macro avg       0.84      0.84      0.84        78
weighted avg       0.95      0.95      0.95        78

accuracy_score- 0.9487179487179487
Model_score - 0.9487179487179487
```

```
C:\Users\Hp\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\linear_model\_logistic.py:444: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

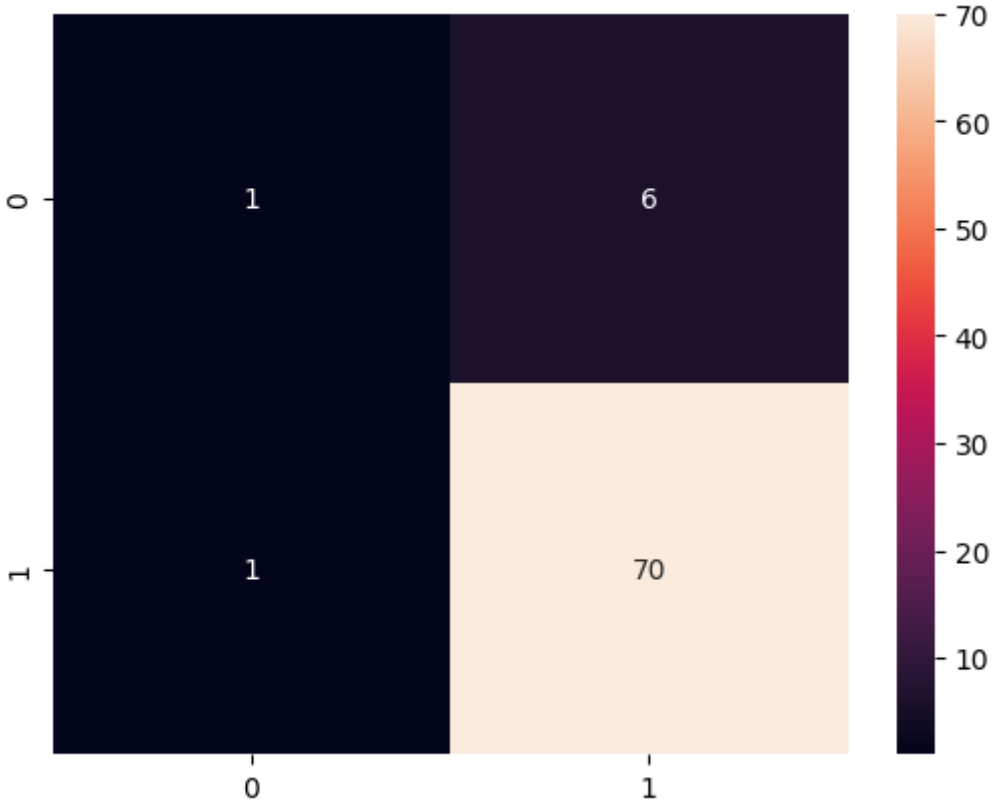


k nearest classifier

In [44]:
```
from sklearn.neighbors import KNeighborsClassifier
lr=KNeighborsClassifier()
lr.fit(x_train, y_train)
```

```
y_pred=lr.predict(x_test)
print("1 denotes 'YES, person have cancer'\n'0 denotes NO,person does not have canc
print (f'first 5 actual_y: {y_test.values[:5]} predicted y: {y_pred[:5]}')
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
print('confusion_matrix')
a=confusion_matrix (y_test,y_pred)
print(a)
print('classification_report')
print(classification_report (y_test,y_pred))
print('accuracy_score-', accuracy_score (y_test,y_pred))
print('Model_score -',lr.score (x_test,y_test))
sns.heatmap(a, annot=True)
plt.show()
```

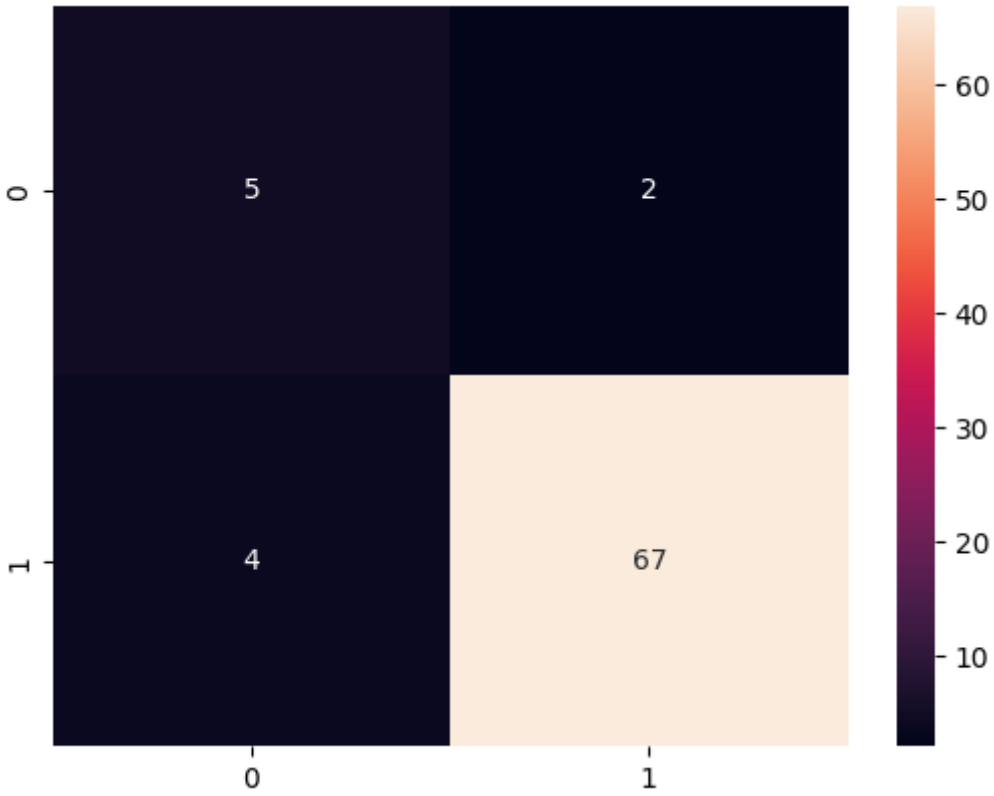
```
1 denotes 'YES, person have cancer'
'0 denotes NO,person does not have cancer
first 5 actual_y: [1 1 1 1 1] predicted y: [1 1 1 1 1]
confusion_matrix
[[ 1  6]
 [ 1 70]]
classification_report
              precision    recall  f1-score   support

           0       0.50      0.14      0.22         7
           1       0.92      0.99      0.95        71

    accuracy                           0.91        78
   macro avg       0.71      0.56      0.59        78
weighted avg       0.88      0.91      0.89        78

accuracy_score- 0.9102564102564102
Model_score - 0.9102564102564102
```



decision tree classifier

In [45]:
```
from sklearn.tree import DecisionTreeClassifier
lr=DecisionTreeClassifier()

lr.fit(x_train, y_train)
y_pred=lr.predict(x_test)
print("1 denotes 'YES, person have cancer'\n'0 denotes NO,person does not have canc
print (f'first 5 actual_y: {y_test.values[:5]} predicted y: {y_pred[:5]}')
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
print('confusion_matrix')
a=confusion_matrix (y_test,y_pred)
print(a)
print('classification_report')
print(classification_report (y_test,y_pred))
print('accuracy_score-', accuracy_score (y_test,y_pred))
print('Model_score -',lr.score (x_test,y_test))
sns.heatmap(a, annot=True)
plt.show()
```

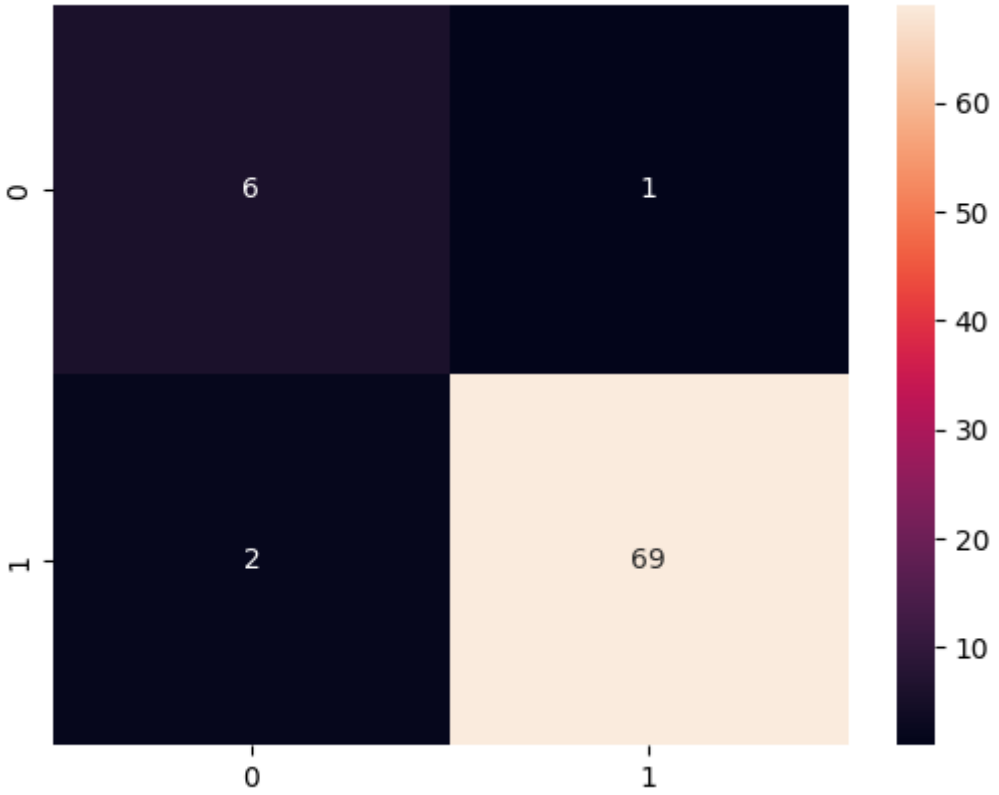
```
1 denotes 'YES, person have cancer'
'0 denotes NO,person does not have cancer
first 5 actual_y: [1 1 1 1 1] predicted y: [1 1 1 1 1]
confusion_matrix
[[ 5  2]
 [ 4 67]]
classification_report
              precision    recall  f1-score   support

           0       0.56      0.71      0.63         7
           1       0.97      0.94      0.96        71

    accuracy                           0.92        78
   macro avg       0.76      0.83      0.79        78
weighted avg       0.93      0.92      0.93        78

accuracy_score- 0.9230769230769231
Model_score - 0.9230769230769231
```



Random forest

In [46]:
```
from sklearn.ensemble import RandomForestClassifier
lr=RandomForestClassifier()
lr.fit(x_train, y_train)
y_pred=lr.predict(x_test)
print("1 denotes 'YES, person have cancer'\n'0 denotes NO,person does not have canc
print (f'first 5 actual_y: {y_test.values[:5]} predicted y: {y_pred[:5]}')
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
print('confusion_matrix')
a=confusion_matrix (y_test,y_pred)
print(a)
print('classification_report')
print(classification_report (y_test,y_pred))
print('accuracy_score-', accuracy_score (y_test,y_pred))
print('Model_score -',lr.score (x_test,y_test))
sns.heatmap(a, annot=True)
plt.show()
```

```
1 denotes 'YES, person have cancer'
'0 denotes NO,person does not have cancer
first 5 actual_y: [1 1 1 1 1] predicted y: [1 1 1 1 1]
confusion_matrix
[[ 6  1]
 [ 2 69]]
classification_report
              precision    recall  f1-score   support

           0       0.75      0.86      0.80         7
           1       0.99      0.97      0.98        71

    accuracy                           0.96        78
   macro avg       0.87      0.91      0.89        78
weighted avg       0.96      0.96      0.96        78

accuracy_score- 0.9615384615384616
Model_score - 0.9615384615384616
```



accuracy of all the model for the given problem is given below

1. linear Regression= 0.2079084725452871
2. Logistic regresssion=0.9487179487179487
3. k nearest neighbour=0.9102564102564102
4. decision tree classifier=0.9230769230769231
5. random forest classifier=0.9615384615384616

from all of the above model the accuracy of the random forest is maximum hence we will use random forest algorithm to detect the lung cancer

In [ ]:

In [ ]: