LUNG CANCER PREDICTION

AIM: PREDICTION OF LUNG CANCER

INTRODUCTION : Lung Cancer is a most effected disease nowadays , most of the people are affecting due to the cause of Lung Cancer, in this dataset we have a huge information about Lung Cancer and it causes , the dataset contains columns like Gender , Age , Smoking , Anxiety , Alcohol Consuming , Coughing ,Shortness of Breathe , Chest Pain , Lung Cancer….

STEPS TO CLASSIFY LUNG CANCER :

1. LOAD THE DATA
2. ANALYSIZE AND VISUALIZE THE DATASET
3. MODEL TRAINING
4. MODEL EVALUATION
5. TESTING THE MODEL

EXPLANATION : Will go through step by step

STEP1 : Loading the data :

To load the data we have to import necessary packages like numpy, pandas, seaborn, matplotlib….etc .

- ❖ Numpy will be used for any computational operations.
- ❖ we'll use matplotlib and seaborn for data visualization .
- ❖ we use pandas for loading data from various sources like local storages , databases , excel , csv files ……etc.
- ❖ Next we load the data using pd.read_csv() . and set the column name as for the iris data information .
- ❖ pd.read_csv reads CSVfiles.csv transfer comma separated value.
- ❖ df.head() only shows the first five rows from the dataset table -All the numeric values.

Importing necessary packages:

```
import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import pandas as pd

get_ipython().run_line_magic('matplotlib', 'inline')
```

Loading the data:

```
df=pd.read_csv('lungcancer.csv')

df
```

	GENDER	AGE	SMOKING	ANXIETY	FATIGUE	ALCOHOL CONSUMING	COUGHING	SHORTNESS OF BREATH	CHEST PAIN	LUNG_CANCER
0	M	69	1	2	2	2	2	2	2	YES
1	M	74	2	1	2	1	1	2	2	YES
2	F	59	1	1	2	1	2	2	2	NO
3	M	63	2	2	1	2	1	1	2	NO
4	F	63	1	1	1	1	2	2	1	NO
...	...	...	...	...	...	...	...	...	...	...
304	F	56	1	1	2	2	2	2	1	YES
305	M	70	2	1	2	2	2	2	2	YES
306	M	58	2	1	1	2	2	1	2	YES
307	M	67	2	2	2	2	2	2	2	YES
308	M	62	1	1	2	2	1	1	1	YES

309 rows × 10 columns

STEP2 : ANALYSING AND VISUALIZE DATASET

- Let’s see information…
- Before that we have done some basic operations like describe (). , info () , head functions and tail functions droping function , indexing and slicing ..etc
- Later we prepare the data in a perfect manner without any null values that means we cleared the data
- we have built some model for the dataset based on that data we do some basic operations and analyzed the data
- After that based on analization we visualized the data with seaborn pairplot,matplotlib….etc

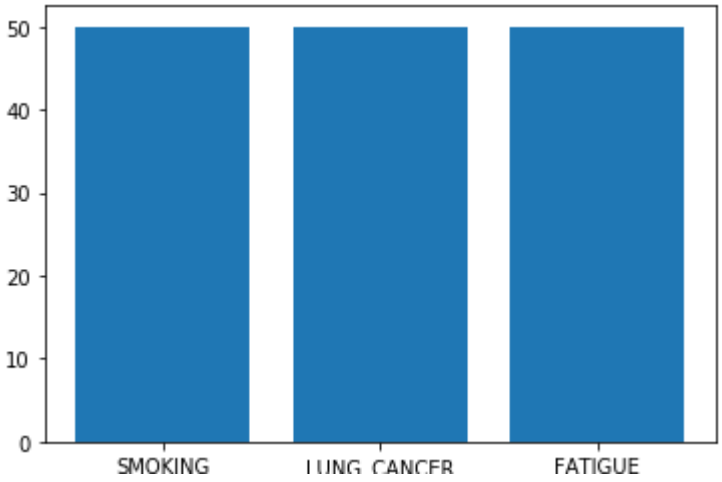
```
import matplotlib.pyplot as plt
```

```
x=['SMOKING','LUNG_CANCER','FATIGUE']

y=[50,50,50]

plt.bar(x,y)
```

Output :



ANALYSING THE DATA:

- Checking the missing values amd filling are removing the empty values
- First we have to check it the dataset contains empty values or not

Code for checking the missing values:

➔ df.isnull()

Output:

	GENDER	AGE	SMOKING	ANXIETY	FATIGUE	ALCOHOL CONSUMING	COUGHING	SHORTNESS OF BREATH	CHEST PAIN	LUNG_CANCER
0	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...
304	False	False	False	False	False	False	False	False	False	False
305	False	False	False	False	False	False	False	False	False	False
306	False	False	False	False	False	False	False	False	False	False
307	False	False	False	False	False	False	False	False	False	False
308	False	False	False	False	False	False	False	False	False	False

309 rows × 10 columns

➔ There is no null values in the given dataset, so we don't need to fill the values.

➔ Next we have check the info of the dataset. To find if there is any object type dataset.

➔ Code for checking info of the dataset.

➔ df.info()

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 309 entries, 0 to 308
Data columns (total 10 columns):
GENDER                  309 non-null object
AGE                     309 non-null int64
SMOKING                 309 non-null int64
ANXIETY                 309 non-null int64
FATIGUE                 309 non-null int64
ALCOHOL CONSUMING       309 non-null int64
COUGHING                309 non-null int64
SHORTNESS OF BREATH     309 non-null int64
CHEST PAIN              309 non-null int64
LUNG_CANCER             309 non-null object
dtypes: int64(8), object(2)
memory usage: 24.3+ KB
```

- We have two columns with string type which is in the type of object.
- We have to change string into float or integer values.
- Code for changing string columns into integer:

```
encoder = LabelEncoder()

df['GENDER'] = encoder.fit_transform(df['GENDER'])

df['LUNG_CANCER'] = encoder.fit_transform(df['LUNG_CANCER'])

df.info()
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 309 entries, 0 to 308
Data columns (total 10 columns):
GENDER                  309 non-null int32
AGE                     309 non-null int64
SMOKING                 309 non-null int64
ANXIETY                 309 non-null int64
FATIGUE                 309 non-null int64
ALCOHOL CONSUMING       309 non-null int64
COUGHING                309 non-null int64
SHORTNESS OF BREATH     309 non-null int64
CHEST PAIN              309 non-null int64
LUNG_CANCER             309 non-null int32
dtypes: int32(2), int64(8)
memory usage: 21.9 KB
```

Step 3 – Model training:

- Using train_test_split we split the whole data into training and testing datasets. Later we'll use the testing dataset to check the accuracy of the model.
  - Here we imported a support vector classifier from the scikit-learn support vector machine.
  - Then, we created an object and named it svn.
  - After that, we feed the training dataset into the algorithm by using the svn.fit() method.

Splitting the data into training and testing using svm :

```
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)
# Support vector machine algorithm
from sklearn.svm import SVC
svn = SVC()
svn.fit(X_train, y_train)
```

Output:

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\svm\base.py:193: FutureWarning: The default value of gamma will change from
'auto' to 'scale' in version 0.22 to account better for unscaled features. Set gamma explicitly to 'auto' or 'scale' to avoid t
his warning.
  "avoid this warning.", FutureWarning)
```

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
    kernel='rbf', max_iter=-1, probability=False, random_state=None,
    shrinking=True, tol=0.001, verbose=False)
```

Step 4 – Model Evaluation:

- Now we predict the classes from the test dataset using our trained model.
- Then we check the accuracy score of the predicted classes.
- accuracy_score() takes true values and predicted values and returns the percentage of accuracy.

```
# Predict from the test dataset
predictions = svn.predict(X_test)
# Calculate the accuracy
from sklearn.metrics import accuracy_score
```

accuracy_score(y_test, predictions)

```
Out[42]: 0.8548387096774194
```

The accuracy score is more than 85%

Classification report:

from sklearn.metrics import classification_report

print(classification_report(y_test, predictions))

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         9
           1       0.85      1.00      0.92        53

    accuracy                           0.85        62
   macro avg       0.43      0.50      0.46        62
weighted avg       0.73      0.85      0.79        62
```

From the classification report we predict how accurate the dataset is

- Precision report tells about ratio of true positives to the sum of true and false positives.
- Recall tells about the ratio of the sum of the true positives and false negatives.
- F1_score tells about the mean between precision and recall.
- If the f1_score is 1.0 ,the expected performance of the model is better.
- Support tells about the number of actual occurences of the class in the dataset.

Step 5 – Testing the model:

d1= float(input("Enter the Gender : "))

```
d2= float(input("Enter the Age : "))
d3= float(input("Enter smoking : "))
d4=float(input("Enter if the person have fatigue:"))
d5= float(input("Enter anxiety : "))
d6=float(input("Enter alcohol consuming rate:"))
d7=float(input("Enter if the person have coughing:"))
d8=float(input("Enter shortness rate of breathe:"))
d9=float(input("Enter if the person have chest pain:"))
data = [[d1, d2, d3, d4,d5,d6,d7,d8,d9]]
prediction = svn.predict(data)
j=0
for i in range(0,8):
    if data[0][i]<0:
        print("Negative values not accepted")
        break
    else:
        j=j+1
if j==8:
    prediction = svn.predict(X_new)
    print("Prediction of Species: {}".format(prediction))
```

Output:

```
Enter the Gender : 1
Enter the Age : 23
Enter smoking : 1
Enter if the person have fatigue:1
Enter anxiety : 1
Enter alcohol consuming rate:1
Enter if the person have coughing:1
Enter shortness rate of breathe:1
Enter if the person have chest pain:1
Prediction of Species: [1]
```

SUMMARY AND ANALASYS IN MY POINT OF VIEW

I worked on the dataset and the model is successful. We got the accurate value from the model. From the output , It look like the model is predicting correctly

Here we learned to train our own supervised machine learning model using Lung cancer Dataset, Classification Project with Machine Learning.

It helps us to learn machine learning, data visualization, model creation etc..

Based on our data set when we are visualizing the data with the help of seaborn we observed that we predicted the data how people will have lung cancer based on the some causes.

Work done by:

G. Jeevana

S180111

CSE-2A