```
import pandas as pd
df=pd.read_csv('/content/lung_cancer_examples (1).csv')
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

```
dir('lung_cancer_examples')
```

```
['__add__',
 '__class__',
 '__contains__',
 '__delattr__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__getnewargs__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__iter__',
 '__le__',
 '__len__',
 '__lt__',
 '__mod__',
 '__mul__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__rmod__',
 '__rmul__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 'capitalize',
 'casefold',
 'center',
 'count',
 'encode',
 'endswith',
 'expandtabs',
 'find',
 'format',
 'format_map',
 'index',
 'isalnum',
 'isalpha',
 'isascii',
 'isdecimal',
 'isdigit',
 'isidentifier',
 'islower',
 'isnumeric',
 'isprintable',
 'isspace',
 'istitle',
 'isupper',
 'join',
 'ljust',
```

```
df
```

	Name	Surname	Age	Smokes	AreaQ	Alkhol	Result
0	John	Wick	35	3	5	4	1
1	John	Constantine	27	20	2	5	1
2	Camela	Anderson	30	0	5	2	0
3	Alex	Telles	28	0	8	1	0
4	Diego	Maradona	68	4	5	6	1
5	Cristiano	Ronaldo	34	0	10	0	0
6	Mihail	Tal	58	15	10	0	0
7	Kathy	Bates	22	12	5	2	0
8	Nicole	Kidman	45	2	6	0	0
9	Ray	Milland	52	18	4	5	1
10	Fredric	March	33	4	8	0	0
11	Yul	Brynner	18	10	6	3	0
12	Joan	Crawford	25	2	5	1	0
13	Jane	Wyman	28	20	2	8	1
14	Anna	Magnani	34	25	4	8	1
15	Katharine	Hepburn	39	18	8	1	0
16	Katharine	Hepburn	42	22	3	5	1
17	Barbra	Streisand	19	12	8	0	0
18	Maggie	Smith	62	5	4	3	1
19	Glenda	Jackson	73	10	7	6	1
20	Jane	Fonda	55	15	1	3	1
21	Maximilian	Schell	33	8	8	1	0
22	Gregory	Peck	22	20	6	2	0
23	Sidney	Poitier	44	5	8	1	0
24	Rex	Harrison	77	3	2	6	1
25	Lee	Marvin	21	20	5	3	0
26	Paul	Scofield	37	15	6	2	0
27	Rod	Steiger	34	12	8	0	0
28	John	Wayne	55	20	1	4	1
29	Gene	Hackman	40	20	2	7	1
30	Marlon	Brando	36	13	5	2	0

```
c=df.drop(["Name","Surname"],axis=1)
```

```
c.head()
```

	Age	Smokes	AreaQ	Alkhol	Result
0	35	3	5	4	1
1	27	20	2	5	1
2	30	0	5	2	0
3	28	0	8	1	0
4	68	4	5	6	1

```
df.shape
```

```
(59, 7)
```

```
x=df.iloc[:,:-1]
y=df.iloc[:,-1]
```

```
feature_cols=["Age","Smokes","AreaQ","Alkhol"]
x=df[feature_cols]
y=df.Result
```

```
#Scale the data from 0 to 1
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
x_scale=scaler.fit_transform(x)
print(x_scale)
```

```
[[0.28813559 0.08823529 0.44444444 0.5       ]
 [0.15254237 0.58823529 0.11111111 0.625     ]
 [0.20338983 0.         0.44444444 0.25      ]
 [0.16949153 0.         0.77777778 0.125     ]
 [0.84745763 0.11764706 0.44444444 0.75      ]
 [0.27118644 0.         1.         0.        ]
 [0.6779661  0.44117647 1.         0.        ]
 [0.06779661 0.35294118 0.44444444 0.25      ]
 [0.45762712 0.05882353 0.55555556 0.        ]
 [0.57627119 0.52941176 0.33333333 0.625     ]
 [0.25423729 0.11764706 0.77777778 0.        ]
 [0.         0.29411765 0.55555556 0.375     ]
 [0.11864407 0.05882353 0.44444444 0.125     ]
 [0.16949153 0.58823529 0.11111111 1.        ]
 [0.27118644 0.73529412 0.33333333 1.        ]
 [0.3559322  0.52941176 0.77777778 0.125     ]
 [0.40677966 0.64705882 0.22222222 0.625     ]
 [0.01694915 0.35294118 0.77777778 0.        ]
 [0.74576271 0.14705882 0.33333333 0.375     ]
 [0.93220339 0.29411765 0.66666667 0.75      ]
 [0.62711864 0.44117647 0.         0.375     ]
 [0.25423729 0.23529412 0.77777778 0.125     ]
 [0.06779661 0.58823529 0.55555556 0.25      ]
 [0.44067797 0.14705882 0.77777778 0.125     ]
 [1.         0.08823529 0.11111111 0.75      ]
 [0.05084746 0.58823529 0.44444444 0.375     ]
 [0.3220339  0.44117647 0.55555556 0.25      ]
 [0.27118644 0.35294118 0.77777778 0.        ]
 [0.62711864 0.58823529 0.         0.5       ]
 [0.37288136 0.58823529 0.11111111 0.875     ]
 [0.30508475 0.38235294 0.44444444 0.25      ]
 [0.6440678  0.58823529 0.22222222 0.375     ]
 [0.49152542 0.44117647 0.         1.        ]
 [0.74576271 0.73529412 0.22222222 0.5       ]
 [0.13559322 0.29411765 0.66666667 0.25      ]
 [0.11864407 0.58823529 0.77777778 0.25      ]
 [0.69491525 0.58823529 0.22222222 0.5       ]
 [0.74576271 0.44117647 0.44444444 0.625     ]
 [0.25423729 0.73529412 0.77777778 0.25      ]
 [0.3220339  0.29411765 0.44444444 0.375     ]
 [0.54237288 0.58823529 0.11111111 0.5       ]
 [0.49152542 0.35294118 0.77777778 0.        ]
 [0.86440678 0.58823529 0.44444444 0.5       ]
 [0.76271186 0.58823529 0.33333333 0.625     ]
 [0.3559322  0.44117647 0.66666667 0.25      ]
 [0.05084746 0.58823529 0.77777778 0.375     ]
 [0.22033898 0.58823529 0.88888889 0.5       ]
 [0.16949153 0.29411765 0.33333333 0.125     ]
 [0.59322034 0.58823529 0.55555556 0.375     ]
 [0.74576271 0.58823529 0.44444444 0.75      ]
 [0.40677966 0.35294118 0.55555556 0.25      ]
 [0.44067797 0.88235294 0.         0.75      ]
 [0.13559322 1.         0.         1.        ]
 [0.28813559 0.58823529 0.44444444 0.125     ]
 [0.13559322 0.38235294 0.55555556 0.125     ]
 [1.         0.58823529 0.44444444 0.5       ]
 [0.96610169 0.44117647 0.22222222 0.625     ]
 [0.42372881 0.88235294 0.22222222 1.        ]
```

```
#Let us split the dataset into training and testing data
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x_scale,y,test_size=0.2,random_state=1)
```

```
#Creating ANN model
model=Sequential()
```

```
input_layer=Dense(4,input_shape=(4,))
```

```
model.add(input_layer)
```

```
hidden_layer1=Dense(4,activation='relu')
```

```
model.add(hidden_layer1)
```

```
hidden_layer2=Dense(4,activation='relu')
model.add(hidden_layer2)
```

```
#Create Output layer
output_layer=Dense(1,activation='sigmoid')
model.add(output_layer)
```

```
model.compile(loss="binary_crossentropy",optimizer='sgd',metrics=['accuracy'])
```

```
df.shape
```

```
(59, 7)
```

```
model.fit(x_train,y_train,epochs=100,batch_size=4)
```

```
Epoch 1/100
12/12 [==============================] - 1s 3ms/step - loss: 0.7007 - accuracy: 0.2979
Epoch 2/100
12/12 [==============================] - 0s 2ms/step - loss: 0.6957 - accuracy: 0.4894
Epoch 3/100
12/12 [==============================] - 0s 2ms/step - loss: 0.6947 - accuracy: 0.5319
Epoch 4/100
12/12 [==============================] - 0s 2ms/step - loss: 0.6938 - accuracy: 0.5319
Epoch 5/100
12/12 [==============================] - 0s 2ms/step - loss: 0.6920 - accuracy: 0.5319
Epoch 6/100
12/12 [==============================] - 0s 2ms/step - loss: 0.6853 - accuracy: 0.5319
Epoch 7/100
12/12 [==============================] - 0s 2ms/step - loss: 0.6688 - accuracy: 0.5319
Epoch 8/100
12/12 [==============================] - 0s 2ms/step - loss: 0.6542 - accuracy: 0.5319
Epoch 9/100
12/12 [==============================] - 0s 2ms/step - loss: 0.6420 - accuracy: 0.5319
Epoch 10/100
12/12 [==============================] - 0s 2ms/step - loss: 0.6239 - accuracy: 0.5319
Epoch 11/100
12/12 [==============================] - 0s 2ms/step - loss: 0.6024 - accuracy: 0.5319
Epoch 12/100
12/12 [==============================] - 0s 2ms/step - loss: 0.5853 - accuracy: 0.5319
Epoch 13/100
12/12 [==============================] - 0s 2ms/step - loss: 0.5710 - accuracy: 0.5745
Epoch 14/100
12/12 [==============================] - 0s 2ms/step - loss: 0.5558 - accuracy: 0.7234
Epoch 15/100
12/12 [==============================] - 0s 2ms/step - loss: 0.5425 - accuracy: 0.7021
Epoch 16/100
12/12 [==============================] - 0s 2ms/step - loss: 0.5272 - accuracy: 0.7660
Epoch 17/100
12/12 [==============================] - 0s 2ms/step - loss: 0.5123 - accuracy: 0.7872
Epoch 18/100
12/12 [==============================] - 0s 2ms/step - loss: 0.4983 - accuracy: 0.8085
Epoch 19/100
12/12 [==============================] - 0s 2ms/step - loss: 0.4848 - accuracy: 0.7872
Epoch 20/100
12/12 [==============================] - 0s 2ms/step - loss: 0.4709 - accuracy: 0.7872
Epoch 21/100
12/12 [==============================] - 0s 3ms/step - loss: 0.4582 - accuracy: 0.8511
Epoch 22/100
12/12 [==============================] - 0s 2ms/step - loss: 0.4442 - accuracy: 0.8723
Epoch 23/100
12/12 [==============================] - 0s 2ms/step - loss: 0.4332 - accuracy: 0.8723
Epoch 24/100
12/12 [==============================] - 0s 2ms/step - loss: 0.4216 - accuracy: 0.8723
Epoch 25/100
12/12 [==============================] - 0s 2ms/step - loss: 0.4101 - accuracy: 0.8723
Epoch 26/100
12/12 [==============================] - 0s 2ms/step - loss: 0.4021 - accuracy: 0.8936
Epoch 27/100
12/12 [==============================] - 0s 2ms/step - loss: 0.3916 - accuracy: 0.8936
Epoch 28/100
12/12 [==============================] - 0s 2ms/step - loss: 0.3811 - accuracy: 0.8936
Epoch 29/100
12/12 [==============================] - 0s 2ms/step - loss: 0.3722 - accuracy: 0.8936
```

```
val_loss,val_acc=model.evaluate(x_test,y_test)
```

```
1/1 [==============================] - 0s 170ms/step - loss: 0.2277 - accuracy: 0.9167
```

```
print(val_loss)
print(val_acc)
```

```
0.22772230207920074
0.9166666865348816
```

```
#Make prediction
lst=[[34,5,8,7]]
```

```
scaled_data=scaler.transform(lst)
print(scaled_data)
```

```
[[0.27118644 0.14705882 0.77777778 0.875     ]]
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but MinMaxScaler was
  warnings.warn(
```

```
result=model.predict(scaled_data)
```

```
1/1 [==============================] - 0s 132ms/step
```

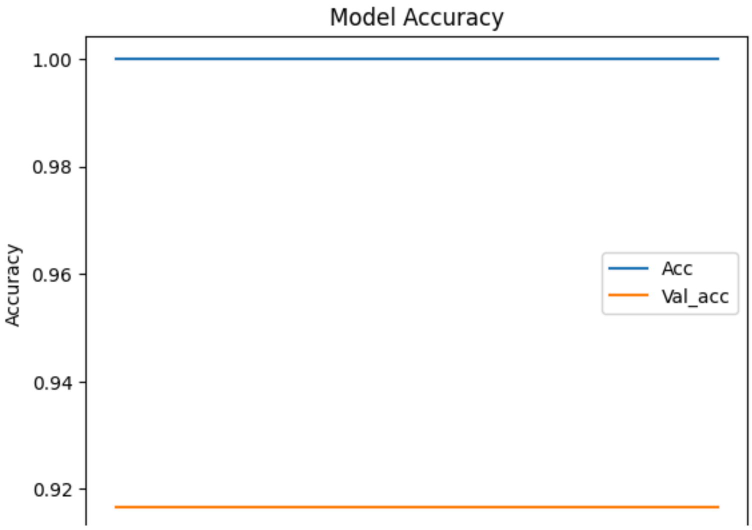
```
if result>=0.5:
  print('1')
else:
  print('0')
```

```
1
```

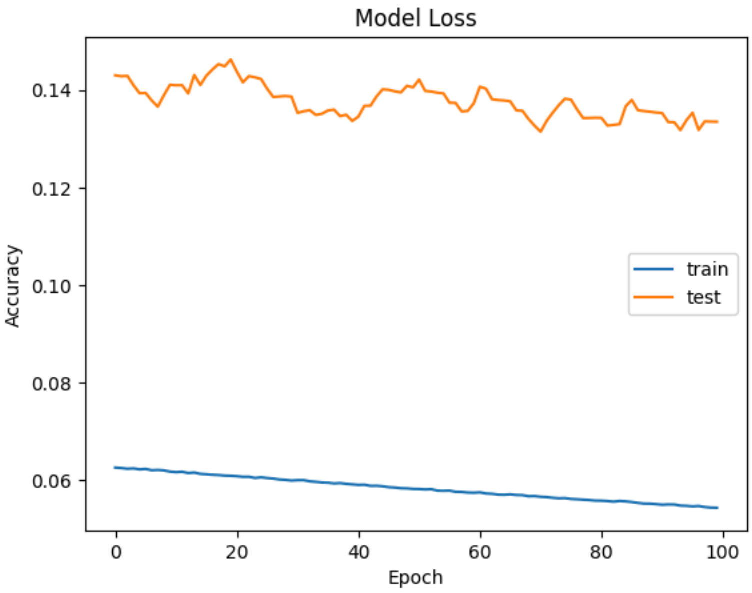
```
#fit() method returns the total history of what happend during tarining
hist=model.fit(x_train,y_train,batch_size=20,epochs=100,validation_data=(x_test,y_test))
```

```
Epoch 1/100
3/3 [==============================] - 0s 46ms/step - loss: 0.0626 - accuracy: 1.0000 - val_loss: 0.1430 - val_accuracy: 0.9167
Epoch 2/100
3/3 [==============================] - 0s 26ms/step - loss: 0.0625 - accuracy: 1.0000 - val_loss: 0.1428 - val_accuracy: 0.9167
Epoch 3/100
3/3 [==============================] - 0s 21ms/step - loss: 0.0623 - accuracy: 1.0000 - val_loss: 0.1429 - val_accuracy: 0.9167
Epoch 4/100
3/3 [==============================] - 0s 27ms/step - loss: 0.0624 - accuracy: 1.0000 - val_loss: 0.1410 - val_accuracy: 0.9167
Epoch 5/100
3/3 [==============================] - 0s 28ms/step - loss: 0.0622 - accuracy: 1.0000 - val_loss: 0.1393 - val_accuracy: 0.9167
Epoch 6/100
3/3 [==============================] - 0s 27ms/step - loss: 0.0623 - accuracy: 1.0000 - val_loss: 0.1394 - val_accuracy: 0.9167
Epoch 7/100
3/3 [==============================] - 0s 21ms/step - loss: 0.0620 - accuracy: 1.0000 - val_loss: 0.1379 - val_accuracy: 0.9167
Epoch 8/100
3/3 [==============================] - 0s 27ms/step - loss: 0.0621 - accuracy: 1.0000 - val_loss: 0.1366 - val_accuracy: 0.9167
Epoch 9/100
3/3 [==============================] - 0s 29ms/step - loss: 0.0620 - accuracy: 1.0000 - val_loss: 0.1389 - val_accuracy: 0.9167
Epoch 10/100
3/3 [==============================] - 0s 27ms/step - loss: 0.0617 - accuracy: 1.0000 - val_loss: 0.1411 - val_accuracy: 0.9167
Epoch 11/100
3/3 [==============================] - 0s 28ms/step - loss: 0.0616 - accuracy: 1.0000 - val_loss: 0.1410 - val_accuracy: 0.9167
Epoch 12/100
3/3 [==============================] - 0s 28ms/step - loss: 0.0617 - accuracy: 1.0000 - val_loss: 0.1410 - val_accuracy: 0.9167
Epoch 13/100
3/3 [==============================] - 0s 28ms/step - loss: 0.0614 - accuracy: 1.0000 - val_loss: 0.1393 - val_accuracy: 0.9167
Epoch 14/100
3/3 [==============================] - 0s 20ms/step - loss: 0.0615 - accuracy: 1.0000 - val_loss: 0.1431 - val_accuracy: 0.9167
Epoch 15/100
3/3 [==============================] - 0s 18ms/step - loss: 0.0613 - accuracy: 1.0000 - val_loss: 0.1410 - val_accuracy: 0.9167
Epoch 16/100
3/3 [==============================] - 0s 19ms/step - loss: 0.0612 - accuracy: 1.0000 - val_loss: 0.1429 - val_accuracy: 0.9167
Epoch 17/100
3/3 [==============================] - 0s 20ms/step - loss: 0.0611 - accuracy: 1.0000 - val_loss: 0.1442 - val_accuracy: 0.9167
Epoch 18/100
3/3 [==============================] - 0s 32ms/step - loss: 0.0610 - accuracy: 1.0000 - val_loss: 0.1453 - val_accuracy: 0.9167
Epoch 19/100
3/3 [==============================] - 0s 19ms/step - loss: 0.0609 - accuracy: 1.0000 - val_loss: 0.1448 - val_accuracy: 0.9167
Epoch 20/100
3/3 [==============================] - 0s 28ms/step - loss: 0.0609 - accuracy: 1.0000 - val_loss: 0.1463 - val_accuracy: 0.9167
Epoch 21/100
3/3 [==============================] - 0s 28ms/step - loss: 0.0608 - accuracy: 1.0000 - val_loss: 0.1438 - val_accuracy: 0.9167
Epoch 22/100
3/3 [==============================] - 0s 19ms/step - loss: 0.0607 - accuracy: 1.0000 - val_loss: 0.1416 - val_accuracy: 0.9167
Epoch 23/100
3/3 [==============================] - 0s 20ms/step - loss: 0.0607 - accuracy: 1.0000 - val_loss: 0.1429 - val_accuracy: 0.9167
Epoch 24/100
3/3 [==============================] - 0s 24ms/step - loss: 0.0604 - accuracy: 1.0000 - val_loss: 0.1426 - val_accuracy: 0.9167
Epoch 25/100
3/3 [==============================] - 0s 20ms/step - loss: 0.0606 - accuracy: 1.0000 - val_loss: 0.1422 - val_accuracy: 0.9167
Epoch 26/100
3/3 [==============================] - 0s 19ms/step - loss: 0.0604 - accuracy: 1.0000 - val_loss: 0.1403 - val_accuracy: 0.9167
Epoch 27/100
3/3 [==============================] - 0s 20ms/step - loss: 0.0603 - accuracy: 1.0000 - val_loss: 0.1386 - val_accuracy: 0.9167
Epoch 28/100
3/3 [==============================] - 0s 19ms/step - loss: 0.0601 - accuracy: 1.0000 - val_loss: 0.1387 - val_accuracy: 0.9167
Epoch 29/100
3/3 [==============================] - 0s 21ms/step - loss: 0.0600 - accuracy: 1.0000 - val_loss: 0.1388 - val_accuracy: 0.9167
```

```
#Visulaize the accuracy in each epoch
import matplotlib.pyplot as plt
plt.plot(hist.history['accuracy'],label='Acc')
plt.plot(hist.history['val_accuracy'],label='Val_acc')
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```



```
#Visulaize the Loss in each epoch
import matplotlib.pyplot as plt
plt.plot(hist.history['loss'],label='train')
plt.plot(hist.history['val_loss'],label='test')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```


Model Loss
Accuracy
train
test
Epoch