```
from numpy import expand_dims
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.preprocessing.image import ImageDataGenerator
from matplotlib import pyplot as plt
```

```
img = load_img('/content/dog.4001.jpg')
```

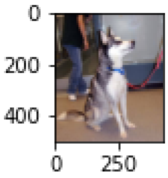
```
data = img_to_array(img)
data = expand_dims(data,0)
```

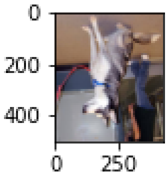
```
datagen = ImageDataGenerator(horizontal_flip=True, vertical_flip=True)
#datagen = ImageDataGenerator(rotation_range=30, fill_mode='nearest')
#datagen = ImageDataGenerator(brightness_range=[0.1,2.5])
#datagen = ImageDataGenerator(width_shift_range=0.2, height_shift_range=0.2, height_shift_
#datagen = ImageDataGenerator(zoom_range=0.25)
```

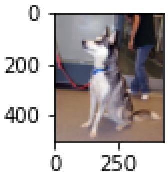
```
iter = datagen.flow(data, batch_size=1)
```

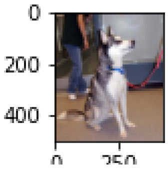
```
for i in range(9):
  plt.subplot(330 + 1 + i)
  batch = iter.next()
  image = batch[0].astype('uint8')
  plt.imshow(image)
  plt.show()
```

Saved successfully! ✕

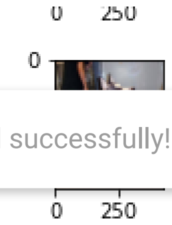




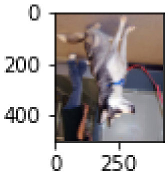


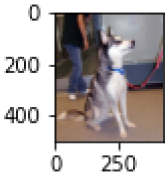


Double-click (or enter) to edit



Saved successfully!





✓ 0s completed at 2:13 PM ● ✕

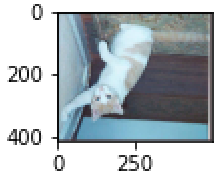
Saved successfully! ✕

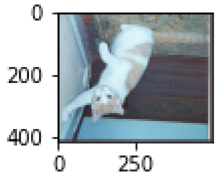
```
from numpy import expand_dims
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.preprocessing.image import ImageDataGenerator
from matplotlib import pyplot as plt
```

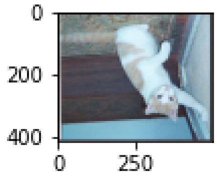
```
img = load_img('/content/cat.4001.jpg')
```

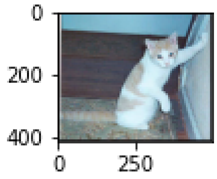
```
data = img_to_array(img)
data = expand_dims(data,0)


datagen = ImageDataGenerator(horizontal_flip=True, vertical_flip=True)
#datagen = ImageDataGenerator(rotation_range=30, fill_mode='nearest')
#datagen = ImageDataGenerator(brightness_range=[0.1,2.5])
#datagen = ImageDataGenerator(width_shift_range=0.2, height_shift_range=0.2, height_shift_
#datagen = ImageDataGenerator(zoom_range=0.25)


iter = datagen.flow(data, batch_size=1)


for i in range(9):
  plt.subplot(330 + 1 + i)
  batch = iter.next()
  image = batch[0].astype('uint8')
  plt.imshow(image)
  plt.show()
```

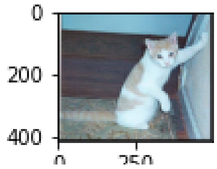
Saving... ×



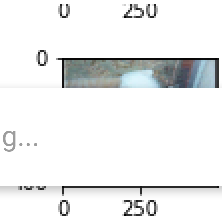




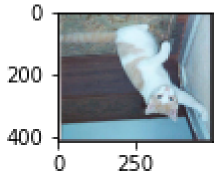


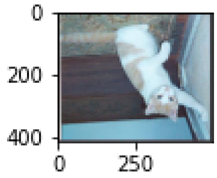


Double-click (or enter) to edit



Saving...





✓ 1s completed at 2:18 PM ● ×

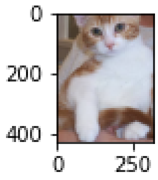
Saving... ×

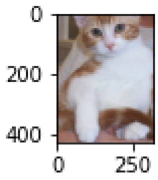
```
from numpy import expand_dims
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.preprocessing.image import ImageDataGenerator
from matplotlib import pyplot as plt
```

```
img = load_img('/content/cat.9.jpg')
```

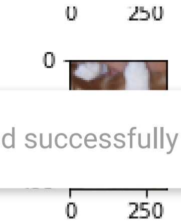
```
data = img_to_array(img)
data = expand_dims(data,0)


datagen = ImageDataGenerator(horizontal_flip=True, vertical_flip=True)
#datagen = ImageDataGenerator(rotation_range=30, fill_mode='nearest')
#datagen = ImageDataGenerator(brightness_range=[0.1,2.5])
#datagen = ImageDataGenerator(width_shift_range=0.2, height_shift_range=0.2, height_shift_
#datagen = ImageDataGenerator(zoom_range=0.25)


iter = datagen.flow(data, batch_size=1)


for i in range(9):
  plt.subplot(330 + 1 + i)
  batch = iter.next()
  image = batch[0].astype('uint8')
  plt.imshow(image)
  plt.show()
```

Saved successfully! ×

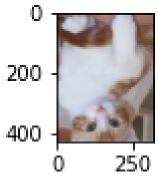




Double-click (or enter) to edit



Saved successfully! ×



Colab paid products - Cancel contracts here

✓ 1s completed at 2:22 PM ● ×

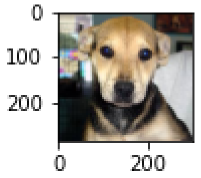
Saved successfully! ×

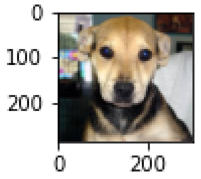
```
from numpy import expand_dims
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.preprocessing.image import ImageDataGenerator
from matplotlib import pyplot as plt
```

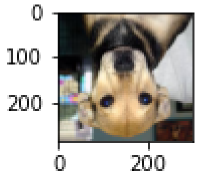
```
img = load_img('/content/dog.4.jpg')
```

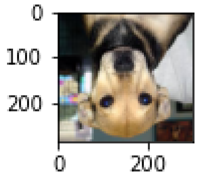
```
data = img_to_array(img)
data = expand_dims(data,0)


datagen = ImageDataGenerator(horizontal_flip=True, vertical_flip=True)
#datagen = ImageDataGenerator(rotation_range=30, fill_mode='nearest')
#datagen = ImageDataGenerator(brightness_range=[0.1,2.5])
#datagen = ImageDataGenerator(width_shift_range=0.2, height_shift_range=0.2, height_shift_
#datagen = ImageDataGenerator(zoom_range=0.25)


iter = datagen.flow(data, batch_size=1)


for i in range(9):
  plt.subplot(330 + 1 + i)
  batch = iter.next()
  image = batch[0].astype('uint8')
  plt.imshow(image)
  plt.show()
```

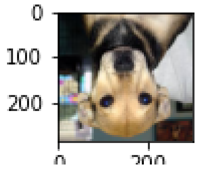
Saving... ×



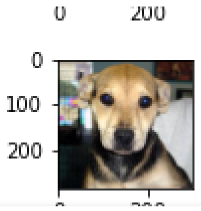




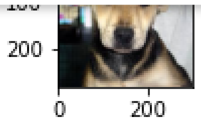


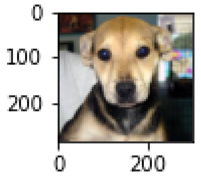


Double-click (or enter) to edit



Saving...





Colab paid products - Cancel contracts here

✓ 1s completed at 2:26 PM

Saving...

```
import pandas as pd
import numpy as np
import io
import seaborn as sns
import matplotlib.pyplot as plt
```

```
data=pd.read_csv('/content/data.csv')
data.head(10)
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smootl
0	842302	M	17.99	10.38	122.80	1001.0	
1	842517	M	20.57	17.77	132.90	1326.0	
2	84300903	M	19.69	21.25	130.00	1203.0	
3	84348301	M	11.42	20.38	77.58	386.1	
4	84358402	M	20.29	14.34	135.10	1297.0	
5	843786	M	12.45	15.70	82.57	477.1	
6	844359	M	18.25	19.98	119.60	1040.0	
7	84458202	M	13.71	20.83	90.20	577.9	
8	844981	M	13.00	21.82	87.50	519.8	
9	84501001	M	12.46	24.04	83.97	475.9	

10 rows × 33 columns

```
del data['Unnamed: 32']
data.head
```

```
566     926954         M        16.60         28.08          108.30      858.1
567     927241         M        20.60         29.33          140.10     1265.0
568      92751         B         7.76         24.54           47.92      181.0

     smoothness_mean  compactness_mean  concavity_mean  concave points_mean  \
0            0.11840           0.27760         0.30010              0.14710

1            0.08474           0.07864         0.08690              0.07017
2            0.10960           0.15990         0.19740              0.12790
3            0.14250           0.28390         0.24140              0.10520
4            0.10030           0.13280         0.19800              0.10430
..               ...               ...             ...                  ...
564          0.11100           0.11590         0.24390              0.13890
565          0.09780           0.10340         0.14400              0.09791
566          0.08455           0.10230         0.09251              0.05302
567          0.11780           0.27700         0.35140              0.15200
568          0.05263           0.04362         0.00000              0.00000
```

```
     ...  radius_worst  texture_worst  perimeter_worst  area_worst  \
0    ...        25.380          17.33           184.60      2019.0   
1    ...        24.990          23.41           158.80      1956.0   
2    ...        23.570          25.53           152.50      1709.0   
3    ...        14.910          26.50            98.87       567.7   
4    ...        22.540          16.67           152.20      1575.0   
..   ...           ...            ...              ...         ...   
564  ...        25.450          26.40           166.10      2027.0   
565  ...        23.690          38.25           155.00      1731.0   
566  ...        18.980          34.12           126.70      1124.0   
567  ...        25.740          39.42           184.60      1821.0   
568  ...         9.456          30.37            59.16       268.6   

     smoothness_worst  compactness_worst  concavity_worst  \
0             0.16220            0.66560           0.7119   
1             0.12380            0.18660           0.2416   
2             0.14440            0.42450           0.4504   
3             0.20980            0.86630           0.6869   
4             0.13740            0.20500           0.4000   
..                ...                ...              ...   
564           0.14100            0.21130           0.4107   
565           0.11660            0.19220           0.3215   
566           0.11390            0.30940           0.3403   
567           0.16500            0.86810           0.9387   
568           0.08996            0.06444           0.0000   

     concave points_worst  symmetry_worst  fractal_dimension_worst  
0                  0.2654          0.4601                  0.11890  
1                  0.1860          0.2750                  0.08902  
2                  0.2430          0.3613                  0.08758  
3                  0.2575          0.6638                  0.17300  
4                  0.1625          0.2364                  0.07678  
..                    ...             ...                      ...  
564                0.2216          0.2060                  0.07115  
565                0.1628          0.2572                  0.06637  
566                0.1418          0.2218                  0.07820  
567                0.2650          0.4087                  0.12400  
568                0.0000          0.2871                  0.07039  

[569 rows x 32 columns]>
```

```
data.columns
```

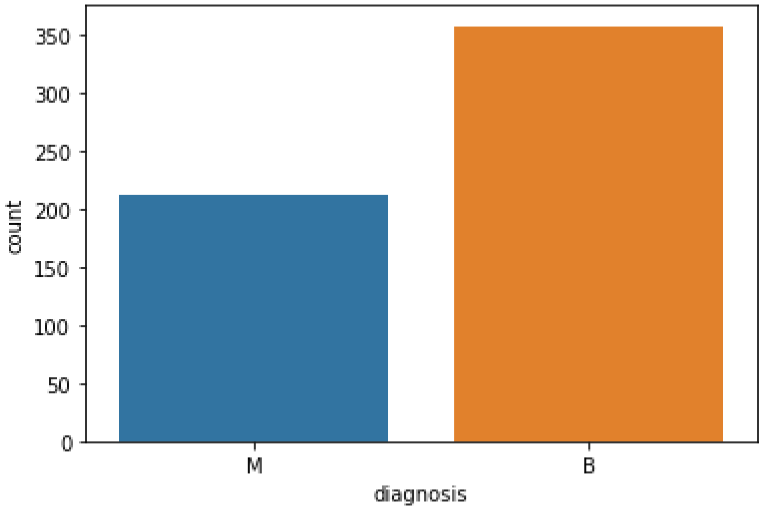
```
Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst'],
      dtype='object')
```

```
data.shape
```

```
(569, 32)
```

```
ax=sns.countplot(data['diagnosis'],label='count')
Benign,Malignanat=data['diagnosis'].value_counts()
print('Benign',Benign)
print('Malignanat',Malignanat)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pas
  FutureWarning
Benign 357
Malignanat 212
```



```
x=data.iloc[:,2:].values
y=data.iloc[:,1].values
```

```
y
```

```
array(['M', 'M', 'M', 'M', 'M', 'M', 'M', 'M', 'M', 'M', 'M', 'M', 'M',
       'M', 'M', 'M', 'M', 'M', 'M', 'B', 'B', 'B', 'M', 'M', 'M', 'M',
       'M', 'M', 'M', 'M', 'M', 'M', 'M', 'M', 'M', 'M', 'M', 'B', 'M',
       'M', 'M', 'M', 'M', 'M', 'M', 'M', 'B', 'M', 'B', 'B', 'B', 'B',
       'B', 'M', 'M', 'B', 'M', 'M', 'B', 'B', 'B', 'B', 'M', 'B', 'M',
       'M', 'B', 'B', 'B', 'B', 'M', 'B', 'M', 'M', 'B', 'M', 'B', 'M',
       'M', 'B', 'B', 'B', 'M', 'M', 'B', 'M', 'M', 'M', 'B', 'B', 'B',
       'M', 'B', 'B', 'M', 'M', 'B', 'B', 'B', 'M', 'M', 'B', 'B', 'B',
       'B', 'M', 'B', 'B', 'M', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B',
       'M', 'M', 'M', 'B', 'M', 'M', 'B', 'B', 'B', 'M', 'M', 'B', 'M',
       'B', 'M', 'M', 'B', 'M', 'M', 'B', 'B', 'M', 'B', 'B', 'M', 'B',
       'B', 'B', 'B', 'M', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B',
       'M', 'B', 'B', 'B', 'B', 'M', 'M', 'B', 'M', 'B', 'B', 'M', 'M',
       'B', 'B', 'M', 'M', 'B', 'B', 'B', 'B', 'M', 'B', 'B', 'M', 'M',
       'M', 'B', 'M', 'B', 'M', 'B', 'B', 'B', 'M', 'B', 'B', 'M', 'M',
       'B', 'M', 'M', 'M', 'M', 'B', 'M', 'M', 'M', 'B', 'M', 'B', 'M',
       'B', 'B', 'M', 'B', 'M', 'M', 'M', 'M', 'B', 'B', 'M', 'M', 'B',
       'B', 'B', 'M', 'B', 'B', 'B', 'B', 'B', 'M', 'M', 'B', 'B', 'M',
       'B', 'B', 'M', 'M', 'B', 'M', 'B', 'B', 'B', 'B', 'M', 'B', 'B',
       'B', 'B', 'B', 'M', 'B', 'M', 'M', 'M', 'M', 'M', 'M', 'M', 'M',
       'M', 'M', 'M', 'M', 'M', 'M', 'B', 'B', 'B', 'B', 'B', 'B', 'M',
       'B', 'M', 'B', 'B', 'M', 'B', 'B', 'M', 'B', 'M', 'M', 'B', 'B',
       'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'M', 'B',
       'B', 'M', 'B', 'M', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B',
       'B', 'B', 'B', 'B', 'B', 'M', 'B', 'B', 'B', 'M', 'B', 'M', 'B',
       'B', 'B', 'B', 'M', 'M', 'M', 'B', 'B', 'B', 'B', 'M', 'B', 'M',
       'B', 'M', 'B', 'B', 'B', 'M', 'B', 'B', 'B', 'B', 'B', 'B', 'B',
       'M', 'M', 'M', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B',
       'B', 'M', 'M', 'B', 'M', 'M', 'M', 'B', 'M', 'M', 'B', 'B', 'B',
       'B', 'B', 'M', 'B', 'B', 'B', 'B', 'B', 'M', 'B', 'B', 'B', 'M',
       'B', 'B', 'M', 'M', 'B', 'B', 'B', 'B', 'B', 'B', 'M', 'B', 'B',
       'B', 'B', 'B', 'B', 'B', 'M', 'B', 'B', 'B', 'B', 'B', 'M', 'B',
       'B', 'M', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B',
       'B', 'M', 'B', 'M', 'M', 'B', 'M', 'B', 'B', 'B', 'B', 'B', 'M',
       'B', 'B', 'M', 'B', 'M', 'B', 'B', 'M', 'B', 'M', 'B', 'B', 'B',
       'B', 'B', 'B', 'B', 'B', 'M', 'M', 'B', 'B', 'B', 'B', 'B', 'B',
       'M', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'M', 'B',
       'B', 'B', 'B', 'B', 'B', 'B', 'M', 'B', 'M', 'B', 'B', 'M', 'B',
       'B', 'B', 'B', 'B', 'M', 'M', 'B', 'M', 'B', 'M', 'B', 'B', 'B',
       'B', 'B', 'M', 'B', 'B', 'M', 'B', 'M', 'B', 'M', 'M', 'B', 'B',
       'B', 'M', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B',
       'M', 'B', 'M', 'M', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B',
       'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B',
       'B', 'B', 'B', 'M', 'M', 'M', 'M', 'M', 'M', 'B'], dtype=object)
```

```
from sklearn.preprocessing import LabelEncoder
LabelEncoder_x_1=LabelEncoder()
y=LabelEncoder_x_1.fit_transform(y)
```

```
y
```

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

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.15)
```

```
x_train
```

```
     array([[1.131e+01, 1.904e+01, 7.180e+01, ..., 6.961e-02, 2.400e-01,
             6.641e-02],
            [1.916e+01, 2.660e+01, 1.262e+02, ..., 1.872e-01, 3.258e-01,
             9.720e-02],
            [1.171e+01, 1.545e+01, 7.503e+01, ..., 7.864e-02, 2.765e-01,
             7.806e-02],
            ...,
            [1.321e+01, 2.525e+01, 8.410e+01, ..., 6.005e-02, 2.444e-01,
             6.788e-02],
            [1.453e+01, 1.934e+01, 9.425e+01, ..., 9.594e-02, 2.471e-01,
             7.463e-02],
            [1.499e+01, 2.520e+01, 9.554e+01, ..., 2.899e-02, 1.565e-01,
             5.504e-02]])
```

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.fit_transform(x_test)
```

```
x_test
```

```
     array([[ 1.29240325,  0.82708707,  1.23191437, ...,  0.48136694,
              0.3640376 , -0.39426536],
            [-0.0293603 , -0.23180814, -0.07646734, ..., -0.21795522,
             -0.52756682, -0.76594409],
            [ 0.23220975, -0.82697173,  0.24659485, ...,  0.50807562,
              0.42740035,  0.09920963],
            ...,
            [-0.53023912, -0.97328278, -0.4656746 , ..., -0.66829317,
             -0.21226171,  1.17822158],
            [-0.61371892, -0.05325907, -0.63628047, ..., -0.33502825,
              0.51640992, -0.24959656],
            [-0.00988168,  0.92132131,  0.03928653, ...,  0.91760865,
              0.32028713,  0.59096918]])
```

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

```
classifier=Sequential()
classifier.add(Dense(units=16,kernel_initializer='uniform',activation='sigmoid',input_dim=

classifier.add(Dense(units=16,kernel_initializer='uniform',activation='sigmoid'))
classifier.add(Dense(units=12,kernel_initializer='uniform',activation='sigmoid'))
classifier.add(Dense(units=8,kernel_initializer='uniform',activation='sigmoid'))
classifier.add(Dense(units=4,kernel_initializer='uniform',activation='sigmoid'))

classifier.add(Dense(units=1,activation='sigmoid'))
classifier.compile(optimizer='Adam',loss='binary_crossentropy',metrics=['accuracy'])
classifier.fit(x_train,y_train,batch_size=100,epochs=7)
```

```
Epoch 1/7
5/5 [==============================] - 1s 4ms/step - loss: 0.7794 - accuracy: 0.3913
Epoch 2/7
5/5 [==============================] - 0s 3ms/step - loss: 0.7736 - accuracy: 0.3913
Epoch 3/7
5/5 [==============================] - 0s 3ms/step - loss: 0.7684 - accuracy: 0.3913
Epoch 4/7
5/5 [==============================] - 0s 3ms/step - loss: 0.7634 - accuracy: 0.3913
Epoch 5/7
5/5 [==============================] - 0s 4ms/step - loss: 0.7585 - accuracy: 0.3913
Epoch 6/7
5/5 [==============================] - 0s 5ms/step - loss: 0.7538 - accuracy: 0.3913
Epoch 7/7
5/5 [==============================] - 0s 3ms/step - loss: 0.7494 - accuracy: 0.3913
<keras.callbacks.History at 0x7f4a65cf6a10>
```

```
y_pred=classifier.predict(x_test)
y_pred
```

```
       [0.5869747 ],
       [0.5869748 ],
       [0.5869747 ],
       [0.5869747 ],
       [0.5869747 ],
       [0.5869747 ],
       [0.58697474],
       [0.58697474],

       [0.5869747 ],
       [0.5869747 ],
       [0.58697474],
       [0.5869747 ],
       [0.5869747 ],
       [0.58697474],
       [0.5869747 ],
       [0.58697474],
       [0.58697474],
       [0.5869747 ],
       [0.58697474],
       [0.58697474],
       [0.58697474],
       [0.5869747 ],
       [0.58697474],
       [0.5869747 ],
       [0.5869747 ],
       [0.58697474],
       [0.5869747 ],
       [0.5869747 ],
       [0.5869747 ],
       [0.58697474],
       [0.5869747 ],
       [0.5869747 ],
       [0.5869747 ],
       [0.5869747 ],
       [0.58697474],
       [0.5869747 ],
       [0.5869747 ],
       [0.5869747 ],
       [0.5869747 ],
       [0.58697474],
       [0.5869747 ],
       [0.5869747 ],
       [0.5869747 ],
       [0.5869747 ],
       [0.58697474],
       [0.5869747 ],
       [0.5869747 ],
       [0.5869747 ],
       [0.58697474],
       [0.5869747 ],
       [0.5869747 ],
       [0.5869747 ],
       [0.5869747 ],
       [0.58697474],
       [0.5869747 ],
       [0.5869747 ],
       [0.58697474]], dtype=float32)
```

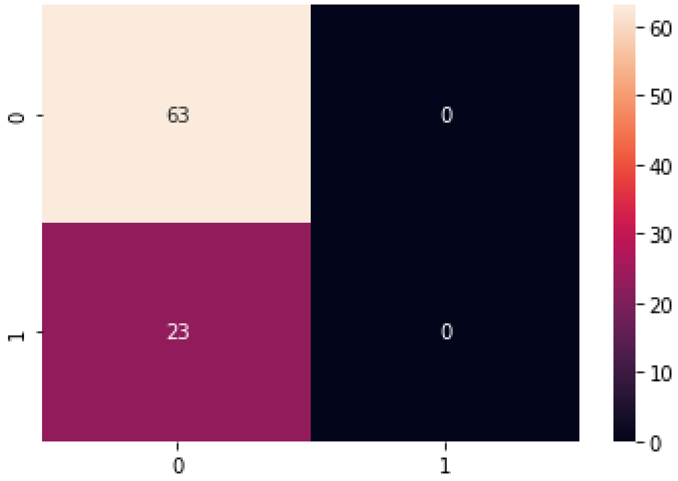
```
y_pred=classifier.predict(x_test)
y_pred=(y_pred>0.6)
```

```
y_pred
```

```
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False]])
```

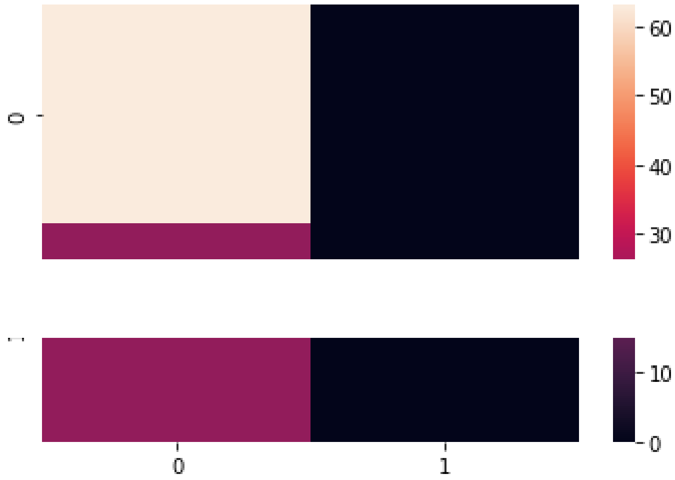
```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
sns.heatmap(cm,annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4a6360c250>
```



```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
sns.heatmap(cm,annot=False)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4a63657890>
```



Colab paid products - Cancel contracts here

✓ 0s completed at 11:07 AM