```
import tensorflow as tf

import pandas as pd

import numpy as np

df = pd.read_csv("C:/Users/Administrator/Desktop/Deep Learning/brstcan.csv")

df.head()

df.info()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\framework\dtypes.py:526:
FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a f
uture version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\framework\dtypes.py:527:
FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a f
uture version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\framework\dtypes.py:528:
FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a f
uture version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\framework\dtypes.py:529:
FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a f
uture version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\framework\dtypes.py:530:
FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a f
uture version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\framework\dtypes.py:535:
FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a f
uture version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 6 columns):
mean_radius        569 non-null float64
mean_texture       569 non-null float64
mean_perimeter     569 non-null float64
mean_area          569 non-null float64
mean_smoothness    569 non-null float64
diagnosis          569 non-null int64
dtypes: float64(5), int64(1)
memory usage: 26.8 KB
```

In [ ]:

```

```

In [34]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 6 columns):
mean_radius        569 non-null float64
mean_texture       569 non-null float64
mean_perimeter     569 non-null float64
mean_area          569 non-null float64
mean_smoothness    569 non-null float64
```

```
diagnosis          569 non-null int64
dtypes: float64(5), int64(1)
memory usage: 26.8 KB
```

In [97]:

```
df.head()
```

Out[97]:

	mean_radius	mean_texture	mean_perimeter	mean_area	mean_smoothness	diagnosis
0	17.99	10.38	122.80	1001.0	0.11840	0
1	20.57	17.77	132.90	1326.0	0.08474	0
2	19.69	21.25	130.00	1203.0	0.10960	0
3	11.42	20.38	77.58	386.1	0.14250	0
4	20.29	14.34	135.10	1297.0	0.10030	0

In [98]:

```
df.isnull().values.any()
```

Out[98]:

```
False
```

In [279]:

```
df.isnull.sum()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-279-69230d699f46> in <module>
----> 1 df.isnull.sum()

AttributeError: 'function' object has no attribute 'sum'
```

In [100]:

```
df.describe()
```

Out[100]:

	mean_radius	mean_texture	mean_perimeter	mean_area	mean_smoothness	diagnosis
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.627417
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.483918
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.000000
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.000000
50%	13.370000	18.840000	86.240000	551.100000	0.095870	1.000000
75%	15.780000	21.800000	104.100000	782.700000	0.105300	1.000000
max	28.110000	39.280000	188.500000	2501.000000	0.163400	1.000000

In [108]:

```
df.groupby('diagnosis').count()
```

Out[108]:

	mean_radius	mean_texture	mean_perimeter	mean_area	mean_smoothness
diagnosis					
0	212	212	212	212	212
1	357	357	357	357	357

In [81]:

```
X_df=df.drop(columns=['diagnosis'])
Y_df = df['diagnosis']
X_df.shape, Y_df.shape
```

Out[81]:

```
((565, 5), (565,))
```

In [60]:

```
X=np.array(X_df)
Y = np.array(Y_df)
```

In [70]:

```
from sklearn.preprocessing import OneHotEncoder
onehot = OneHotEncoder(sparse =False)
y= onehot.fit_transform.reshape(y.shape[0],1)
y.shape
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-70-ccda7d1403a8> in <module>
      1 from sklearn.preprocessing import OneHotEncoder
      2 onehot = OneHotEncoder(sparse =False)
----> 3 y= onehot.fit_transform.reshape(y.shape[0],1)
      4 y.shape

AttributeError: 'function' object has no attribute 'reshape'
```

In [109]:

```
df.drop([9.465],axis =0, inplace=True)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-109-16e24d3117d7> in <module>
----> 1 df.drop([9.465],axis =0, inplace=True)

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\frame.py in drop(self, labels,
axis, index, columns, level, inplace, errors)
   3695                                           index=index, columns=columns,
   3696                                           level=level, inplace=inplace,
-> 3697                                           errors=errors)
   3698
   3699     @rewrite_axis_style_signature('mapper', [('copy', True),

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\generic.py in drop(self, labels
, axis, index, columns, level, inplace, errors)
   3109         for axis, labels in axes.items():
   3110             if labels is not None:
-> 3111                 obj = obj._drop_axis(labels, axis, level=level, errors=errors)
   3112
   3113         if inplace:

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\generic.py in _drop_axis(self,
labels, axis, level, errors)
   3141                 new_axis = axis.drop(labels, level=level, errors=errors)
   3142             else:
```

```
-> 3143                 new_axis = axis.drop(labels, errors=errors)
   3144             result = self.reindex(**{axis_name: new_axis})
   3145

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in drop(self, l
abels, errors)
   4402             if errors != 'ignore':
   4403                 raise KeyError(
-> 4404                     '{} not found in axis'.format(labels[mask]))
   4405             indexer = indexer[~mask]
   4406         return self.delete(indexer)

KeyError: '[9.465] not found in axis'
```

In [114]:

```
df.head()

df.groupby('diagnosis').count()
```

Out[114]:

	mean_radius	mean_texture	mean_perimeter	mean_area	mean_smoothness
diagnosis					
0	212	212	212	212	212
1	357	357	357	357	357

In [160]:

```
df = pd.read_csv("C:/Users/Administrator/Desktop/Deep Learning/brstcanup.csv")

df.head()
```

Out[160]:

	mean_radius	mean_texture	mean_perimeter	mean_area	mean_smoothness	diagnosis	Unnamed: 6	Unnamed: 7	Unnamed: 8
0	17.99	10.38	122.80	1001.0	0.11840	0	NaN	NaN	NaN
1	20.57	17.77	132.90	1326.0	0.08474	0	NaN	NaN	NaN
2	19.69	21.25	130.00	1203.0	0.10960	0	NaN	NaN	NaN
3	11.42	20.38	77.58	386.1	0.14250	0	NaN	NaN	NaN
4	20.29	14.34	135.10	1297.0	0.10030	0	NaN	NaN	NaN

In [161]:

```
df.info()

df.isnull().values.any()

df.isnull.sum()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 424 entries, 0 to 423
Data columns (total 9 columns):
mean_radius         424 non-null float64
mean_texture        424 non-null float64
mean_perimeter      424 non-null float64
mean_area           424 non-null float64
mean_smoothness     424 non-null float64
diagnosis           424 non-null int64
Unnamed: 6          0 non-null float64
Unnamed: 7          0 non-null float64
Unnamed: 8          2 non-null float64
dtypes: float64(8), int64(1)
memory usage: 29.9 KB
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-161-85044d08ac8f> in <module>
      1 df.info()
      2 df.isnull().values.any()
----> 3 df.isnull.sum()

AttributeError: 'function' object has no attribute 'sum'
```

In [154]:

```
df.dropna(inplace=True)

df.isnull().values.any()

df.groupby('diagnosis').count()
```

Out[154]:

	mean_radius	mean_texture	mean_perimeter	mean_area	mean_smoothness	Unnamed: 6	Unnamed: 7	Unnamed: 8
diagnosis								

In [164]:

```
df.groupby('diagnosis').count()


df.info()

df.isnull().values.any()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 424 entries, 0 to 423
Data columns (total 9 columns):
mean_radius         424 non-null float64
mean_texture        424 non-null float64
mean_perimeter      424 non-null float64
mean_area           424 non-null float64
mean_smoothness     424 non-null float64
diagnosis           424 non-null int64
Unnamed: 6          0 non-null float64
Unnamed: 7          0 non-null float64
Unnamed: 8          2 non-null float64
dtypes: float64(8), int64(1)
memory usage: 29.9 KB
```

Out[164]:

```
True
```

In [197]:

```
df = pd.read_csv("C:/Users/Administrator/Desktop/Deep Learning/brstcrup2.csv")

df.head()
```

Out[197]:

	mean_radius	mean_texture	mean_perimeter	mean_area	mean_smoothness	diagnosis
0	17.99	10.38	122.80	1001.0	0.11840	0
1	20.57	17.77	132.90	1326.0	0.08474	0
2	19.69	21.25	130.00	1203.0	0.10960	0
3	11.42	20.38	77.58	386.1	0.14250	0
4	20.29	14.34	135.10	1297.0	0.10030	0

In [199]:

```
df.groupby('diagnosis').count()
```

Out[199]:

	mean_radius	mean_texture	mean_perimeter	mean_area	mean_smoothness
diagnosis					
0	212	212	212	212	212
1	212	212	212	212	212

In [174]:

```
df.info()

df.isnull().values.any()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 424 entries, 0 to 423
Data columns (total 6 columns):
mean_radius        424 non-null float64
mean_texture       424 non-null float64
mean_perimeter     424 non-null float64
mean_area          424 non-null float64
mean_smoothness    424 non-null float64
diagnosis          424 non-null int64
dtypes: float64(5), int64(1)
memory usage: 20.0 KB
```

Out[174]:

```
False
```

In [176]:

```
df.describe()
```

Out[176]:

	mean_radius	mean_texture	mean_perimeter	mean_area	mean_smoothness	diagnosis
count	424.000000	424.000000	424.000000	424.000000	424.000000	424.000000
mean	14.853038	19.458373	97.044552	724.004953	0.097746	0.500000
std	3.663706	4.176057	25.280552	375.477080	0.013563	0.500591
min	6.981000	9.710000	43.790000	143.500000	0.062510	0.000000
25%	12.160000	16.490000	78.032500	450.650000	0.087818	0.000000
50%	14.005000	19.055000	91.170000	602.900000	0.097795	0.500000
75%	17.367500	21.885000	114.250000	936.775000	0.106300	1.000000
max	28.110000	39.280000	188.500000	2501.000000	0.144700	1.000000

In [200]:

```
X_df=df.drop(columns=['diagnosis'])

Y_df = df['diagnosis']

X_df.shape, Y_df.shape
```

Out[200]:

```
((424, 5), (424,))
```

In [201]:

```
X=np.array(X_df)

Y=np.array(Y_df)
```

```
from sklearn.preprocessing import OneHotEncoder

onehot = OneHotEncoder(sparse =False)

y= onehot.fit_transform(Y.reshape(Y.shape[0],1))

y.shape
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\preprocessing\_encoders.py:368: Fut
ureWarning: The handling of integer data will change in version 0.22. Currently, the c
ategories are determined based on the range [0, max(values)], while in the future they
will be determined based on the unique values.
If you want the future behaviour and silence this warning, you can specify "categories
='auto'".
In case you used a LabelEncoder before this OneHotEncoder to convert the categories to
integers, then you can now use the OneHotEncoder directly.
  warnings.warn(msg, FutureWarning)
```

Out[201]:

```
(424, 2)
```

In [202]:

```
from sklearn.preprocessing import MinMaxScaler

scale = MinMaxScaler()

x = scale.fit_transform(X)
```

In [149]:

```
import tensorflow as tf
```

In [205]:

```
def model_keras(X):


    model_k = tf.keras.models.Sequential()
    model_k.add(tf.keras.layers.BatchNormalization(input_shape=(5,)))


    model_k .add(tf.keras.layers.Dense(1000, activation ='relu'))
    model_k .add(tf.keras.layers.Dropout(0,5))
    model_k .add(tf.keras.layers. BatchNormalization())


    model_k .add(tf.keras.layers. Dense(500, activation='relu'))
    model_k .add(tf.keras.layers. BatchNormalization())
    model_k .add(tf.keras.layers.Dropout(0,2))


    model_k .add(tf.keras.layers. Dense(50, activation='relu'))
    model_k .add(tf.keras.layers. BatchNormalization())
```

```
    model_k .add(tf.keras.layers.Dropout(0,2))


    model_k .add(tf.keras.layers. Dense(2))
    model_k. add(tf.keras.layers.Activation('sigmoid'))


    model_k. compile(tf.keras.optimizers.Adam(lr=0.0001), loss='binary_crossentropy',
metrics=['accuracy'])
    return model_k
```

In [206]:

```
model_keras = model_keras(X)
model_keras.fit(X,y,epochs = 5,verbose=1, validation_split=0.1)
```

```
WARNING:tensorflow:From C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\o
ps\resource_variable_ops.py:435: colocate_with (from tensorflow.python.framework.ops)
is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\k
eras\layers\core.py:143: calling dropout (from tensorflow.python.ops.nn_ops) with keep
_prob is deprecated and will be removed in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`
.
Train on 381 samples, validate on 43 samples
WARNING:tensorflow:From C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\o
ps\math_ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is deprecated and
will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Epoch 1/5
381/381 [==============================] - 2s 5ms/sample - loss: 0.3665 - acc: 0.8373
- val_loss: 0.5862 - val_acc: 0.6744
Epoch 2/5
381/381 [==============================] - 0s 504us/sample - loss: 0.2243 - acc: 0.896
3 - val_loss: 0.5792 - val_acc: 0.6744
Epoch 3/5
381/381 [==============================] - 0s 467us/sample - loss: 0.2018 - acc: 0.919
9 - val_loss: 0.5824 - val_acc: 0.6744
Epoch 4/5
381/381 [==============================] - 0s 491us/sample - loss: 0.2353 - acc: 0.901
6 - val_loss: 0.5792 - val_acc: 0.6744
Epoch 5/5
381/381 [==============================] - 0s 509us/sample - loss: 0.1801 - acc: 0.933
1 - val_loss: 0.5655 - val_acc: 0.6744
```

Out[206]:

```
<tensorflow.python.keras.callbacks.History at 0x1de8b927e10>
```

In [207]:

```
val_loss = model_keras.history.history['val_loss']
tra_loss = model_keras.history.history['loss']
```

```
val_acc = model_keras.history.history['val_acc']
tra_acc = model_keras.history.history['acc']
```

In [208]:

```
import matplotlib.pyplot as plt
plt.plot(val_acc)

plt.plot(tra_loss)
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title(' Loss Curve')
plt.legend(['Validation Loss','Taining Loss'])
plt.show()
```

In [209]:

```
```

In [210]:

```
from sklearn.preprocessing import MinMaxScaler
test_df=pd.read_csv("C:/Users/Administrator/Desktop/Deep Learning/brstcrup2.csv")
test_df.head()
y_test= np.array(test_df['diagnosis'])
x_test= np.array(test_df.drop(columns='diagnosis'))
x_test.shape, y_test.shape
```

Out[210]:

```
((424, 5), (424,))
```

In [245]:

```
X_test = MinMaxScaler().fit_transform(x_test)
y_test[0:10]
```

Out[245]:

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int64)
```

In [251]:

```
y_test= OneHotEncoder(sparse=False).fit_transform(y_test.reshape(y_test.shape[0],1))
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\preprocessing\_encoders.py:368: Fut
ureWarning: The handling of integer data will change in version 0.22. Currently, the c
```

```
ategories are determined based on the range [0, max(values)], while in the future they
will be determined based on the unique values.
If you want the future behaviour and silence this warning, you can specify "categories
='auto'".
In case you used a LabelEncoder before this OneHotEncoder to convert the categories to
integers, then you can now use the OneHotEncoder directly.
  warnings.warn(msg, FutureWarning)
```

In [263]:

```
y_prd = model_keras.predict(X_test)
```

In [264]:

```
from sklearn.metrics import classification_report, confusion_matrix

y_prd = np.argmax(y_prd,1)

y_actual = np.argmax(y_test, 1)

print(classification_report(y_actual, y_prd))

print(confusion_matrix(y_actual, y_prd))

print(confusion_matrix(y_actual, y_prd))
```

```
              precision    recall  f1-score   support

           0       1.00      0.03      0.06       212
           1       0.51      1.00      0.67       212

   micro avg       0.52      0.52      0.52       424
   macro avg       0.75      0.52      0.37       424
weighted avg       0.75      0.52      0.37       424

[[  7 205]
 [  0 212]]
[[  7 205]
 [  0 212]]
```

In [266]:

```
print(confusion_matrix(y_actual, y_prd))
```

```
[[  7 205]
 [  0 212]]
```

In [67]:

```
import glob

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from skimage import transform

import cv2 as cv2

import seaborn as sns
```

```
---------------------------------------------------------------------------
ImportError                               Traceback (most recent call last)
<ipython-input-67-5768daafe176> in <module>
      3 import pandas as pd
      4 import matplotlib.pyplot as plt
```

```
----> 5 from skimage import transform
      6 import cv2 as cv2
      7 import seaborn as sns

C:\ProgramData\Anaconda3\lib\site-packages\skimage\__init__.py in <module>
    174                                dtype_limits)
    175
--> 176     from .util.lookfor import lookfor
    177     from .data import data_dir
    178

C:\ProgramData\Anaconda3\lib\site-packages\skimage\util\__init__.py in <module>
      6 from .apply_parallel import apply_parallel
      7
----> 8 from .arraycrop import crop
      9 from ._regular_grid import regular_grid, regular_seeds
     10 from .unique import unique_rows

C:\ProgramData\Anaconda3\lib\site-packages\skimage\util\arraycrop.py in <module>
      6
      7 import numpy as np
----> 8 from numpy.lib.arraypad import _validate_lengths
      9
     10

ImportError: cannot import name '_validate_lengths' from 'numpy.lib.arraypad' (C:\Prog
ramData\Anaconda3\lib\site-packages\numpy\lib\arraypad.py)
```

In [ ]:

```
df = pd.read_csv("C:/Users/Administrator/Desktop/Deep Learning/indexcsc.csv")

df.head()
```

In [ ]:

```
df_count= df.groupby('id').count().sort_values(by = df.columns[0], ascending = False)

df_count.columns = ['urlcount']

df_count
```

In [ ]:

```
df_filter = df_count[(df_count['url'])>=50]

df_filter
```

In [1]:

```
import tensorflow_datasets as tfds

import tensorflow as tf
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-1-5af7b16c2677> in <module>
----> 1 import tensorflow_datasets as tfds
      2 import tensorflow as tf

ModuleNotFoundError: No module named 'tensorflow_datasets'
```

In [69]:

```
import numpy as np
```

```
from PIL import Image

im = Image.open("C:/Users/Administrator/Desktop/Deep Learning/aircraft/test/000010.jpg
")

k=np.array(im)

np.savetxt('filename',k,delimiter=',')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-69-c0422e101ba9> in <module>
      3 im = Image.open("C:/Users/Administrator/Desktop/Deep Learning/aircraft/test/00
0010.jpg")
      4 k=np.array(im)
----> 5 np.savetxt('filename',k,delimiter=',')

<__array_function__ internals> in savetxt(*args, **kwargs)

C:\ProgramData\Anaconda3\lib\site-packages\numpy\lib\npyio.py in savetxt(fname, X, fmt
, delimiter, newline, header, footer, comments, encoding)
   1380         if X.ndim == 0 or X.ndim > 2:
   1381             raise ValueError(
-> 1382                 "Expected 1D or 2D array, got %dD array instead" % X.ndim)
   1383         elif X.ndim == 1:
   1384             # Common case -- 1d array of numbers

ValueError: Expected 1D or 2D array, got 3D array instead
```

In [10]:

```
import cv2
```

In [24]:

```
coun= 1

while coun<=10:

    table = coun*2

    coun = coun+1

    print(table)
```

```
2
4
6
8
10
12
14
16
18
20
```

In [30]:

```
count = 1

while count<=10:

 table = 2*count

 count = count+1

 print (table)
```

```
for i in range(10):
    val = i+1
    table = 2*val
    print(table)

for i in range (14):
    table =2*(i+1)
    print(table)
```

```
2
4
6
8
10
12
14
16
18
20
2
4
6
8
10
12
14
16
18
20
2
4
6
8
10
12
14
16
18
20
22
24
26
28
```

In [45]:

```
marks = 90
if (marks==0):
    print("fail")
elif(marks>85):
    print('grade B')
```

```
else:
     print('A')
```

```
grade B
```

In [50]:

```
def table(a):
    for i in range(10):
        table = a*(i+1)
        print(table)
```

In [53]:

```
table(12)
```

```
12
24
36
48
60
72
84
96
108
120
```

In [65]:

```
img =imread("C:/Users/Administrator/Desktop/Deep Learning/aircraft/test/000010.jpg")
io.imshow(img)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-65-e62d4a5d1c90> in <module>
      1
----> 2 img =imread("C:/Users/Administrator/Desktop/Deep Learning/aircraft/test/000010
.jpg")
      3 io.imshow(img)

NameError: name 'imread' is not defined
```

In [20]:

```
import numpy as np


from skimage import io
img = io.imread("C:/Users/Administrator/Desktop/Deep Learning/aircraft/test/000010.jpg
")
io.imshow(img)


# image show
```

Out[20]:

```
<matplotlib.image.AxesImage at 0x2133d4165f8>
```

In [13]:

```
import numpy as np
from matplotlib import pyplot as plt
random_image = np.random.random([500,500])
plt.imshow(random_image,cmap='gray')
plt.colorbar();
```

In [21]:

```
from skimage import data
coins =data.coins()
print('Type', type(coins))
print('dtype',coins.dtype)
print('shap',coins.shape)
plt.imshow(coins,cmap='gray');
```

```
Type <class 'numpy.ndarray'>
dtype uint8
shap (303, 384)
```

In [23]:

```
#Getting Image Resolution
from skimage import io
img = io.imread("C:/Users/Administrator/Desktop/Deep Learning/aircraft/test/000010.jpg
")
img.shape
```

Out[23]:

```
(72, 117, 3)
```

In [27]:

```
#Getting Pixel Values
from skimage import io
import pandas as pd
img = io.imread("C:/Users/Administrator/Desktop/Deep Learning/aircraft/test/000010.jpg
")
df = pd.DataFrame(img.flatten())
filepath = 'pixel_values1.xlsx'
df.to_excel(filepath, index=False)
print(df)
```

```
         0
0      117
```

```
1      138
2      167
3      117
4      138
5      167
6      117
7      138
8      167
9      117
10     138
11     167
12     117
13     138
14     167
15     117
16     138
17     167
18     117
19     138
20     167
21     117
22     138
23     167
24     117
25     138
26     167
27     117
28     138
29     167
...    ...
25242  122
25243  124
25244  123
25245   94
25246   96
25247   95
25248  146
25249  150
25250  149
25251  140
25252  144
25253  143
25254  136
25255  140
25256  139
25257  128
25258  128
25259  128
25260   89
25261   89
25262   89
25263  102
25264  102
25265  102
25266  111
25267  111
25268  111
25269   72
25270   72
25271   72

[25272 rows x 1 columns]
```

In [35]:

```
from skimage import data
from skimage import io
from skimage import color
from pylab import *
#read image

img = io.imread("C:/Users/Administrator/Desktop/Deep Learning/aircraft/test/000010.jpg
")

#Convert to HSV
img_hsv = color.rgb2hsv(img)

#Convert back to RGB
img_rgb = color.hsv2rgb(img_hsv)

 #Show both figures
figure(0)
io.imshow(img_hsv)
figure(1)
io.imshow(img_rgb)
```

Out[35]:

```
<matplotlib.image.AxesImage at 0x2133f04f320>
```

In [37]:

```
#Convert to XYZ
img_xyz = color.rgb2xyz(img)

#Convert back to RGB
img_rgb = color.xyz2rgb(img_xyz)

#Show both figures
figure(0)
io.imshow(img_xyz)
figure(1)
io.imshow(img_rgb)
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or
[0..255] for integers).
```

Out[37]:

```
<matplotlib.image.AxesImage at 0x2133d441be0>
```

In [39]:

```
#Convert to LAB
img_lab = color.rgb2lab(img)

#Convert back to RGB
img_rgb = color.lab2rgb(img_lab)

#Show both figures
figure(0)
io.imshow(img_lab)
figure(1)
io.imshow(img_rgb)
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or
[0..255] for integers).
```

Out[39]:

```
<matplotlib.image.AxesImage at 0x2133f3013c8>
```

In [41]:

```
#Convert to YUV
img_yuv = color.rgb2yuv(img)

#Convert back to RGB
img_rgb = color.yuv2rgb(img_yuv)

#Show both figures
figure(0)
io.imshow(img_yuv)
figure(1)
io.imshow(img_rgb)
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or
[0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or
[0..255] for integers).
```

Out[41]:

```
<matplotlib.image.AxesImage at 0x2133f56f908>
```

In [43]:

```
#Convert to YIQ
img_yiq = color.rgb2yiq(img)

#Convert back to RGB
img_rgb = color.yiq2rgb(img_yiq)

#Show both figures
figure(0)
io.imshow(img_yiq)
figure(1)
io.imshow(img_rgb)
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or
[0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or
[0..255] for integers).
```

Out[43]:

```
<matplotlib.image.AxesImage at 0x2133ef3c4a8>
```

In [45]:

```
#Convert to YPbPr
img_ypbpr= color.rgb2ypbpr(img)

#Convert back to RGB
img_rgb= color.ypbpr2rgb(img_ypbpr)

#Show both figures
figure(0)
io.imshow(img_ypbpr)
figure(1)
io.imshow(img_rgb)
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or
[0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or
[0..255] for integers).
```

Out[45]:

```
<matplotlib.image.AxesImage at 0x2133f6de630>
```

In [47]:

```
#Convert to YPbPr
img_ypbpr= color.rgb2ypbpr(img)


#Convert back to RGB
img_rgb= color.ypbpr2rgb(img_ypbpr)
io.imsave("aircraft_ypbpr.jpg", img_ypbpr)
```

```
WARNING:root:Lossy conversion from float64 to uint8. Range [-0.1730581333333333, 0.999
9999999999999]. Convert image to uint8 prior to saving to suppress this warning.
```

In [52]:

```
from skimage import io
from skimage import draw


img = io.imread("C:/Users/Administrator/Desktop/Deep Learning/aircraft/test/000010.jpg
")


x,y = draw.line(0,0,50,50)
img[x, y] = 0
io.imshow(img)
```

Out[52]:

```
<matplotlib.image.AxesImage at 0x21340ba4898>
```

In [77]:

```
from skimage import io
from skimage import data
img = io.imread("C:/Users/Administrator/Desktop/Deep Learning/aircraft/test/000010.jpg
")
def rectangle(x, y, w, h):
rr,cc = [x, x + w, x + w, x], [y, y, y + h, y + h]
return (draw.polygon(rr, cc))
rr, cc = rectangle(10, 10, 500,500)
img[rr, cc] = 1
io.imshow(img)
```

```
  File "<ipython-input-77-beeb09beb9f5>", line 5
    rr,cc = [x, x + w, x + w, x], [y, y, y + h, y + h]
    ^
IndentationError: expected an indented block
```

In [81]:

```
from skimage import io
from skimage import data
```

```
img = io.imread("C:/Users/Administrator/Desktop/Deep Learning/aircraft/test/000010.jpg
")


#Define circle coordinates and radius
x, y = draw.circle(50,50, 5)


#Draw circle
img[x, y] = 1


#Show image
io.imshow(img)
```

Out[81]:

```
<matplotlib.image.AxesImage at 0x2133d3659b0>
```

In [85]:

```
from skimage import io
from skimage import data
img = io.imread("C:/Users/Administrator/Desktop/Deep Learning/aircraft/test/000010.jpg
")



#Define Bezier curve coordinates
x, y = draw.bezier_curve(0,0, 50, 50, 9,10,10)


#Draw Bezier curve
img[x, y] = 1


#Show image
io.imshow(img)
```

Out[85]:

```
<matplotlib.image.AxesImage at 0x2133f5a0c88>
```

In [89]:

```
from skimage import io
from skimage import draw


#Load image
img = io.imread("C:/Users/Administrator/Desktop/Deep Learning/Tiger.jpg")
```

```
#Define Bezier curve coordinates
x, y = draw.bezier_curve(0,0, 50, 50, 90,120,10)



#Draw Bezier curve
img[x, y] = 1

#Show image
io.imshow(img)
```

Out[89]:

```
<matplotlib.image.AxesImage at 0x2133f6765c0>
```

In [95]:

```
from skimage import exposure
from skimage import io
from pylab import *
img = io.imread("C:/Users/Administrator/Desktop/Deep Learning/Tiger.jpg")
gamma_corrected1 = exposure.adjust_gamma(img, 0.5)
gamma_corrected2 = exposure.adjust_gamma(img, 5)
gamma_corrected3 = exposure.adjust_gamma(img,2)
gamma_corrected4 = exposure.adjust_gamma(img, 0.3)
figure(0)
io.imshow(gamma_corrected1)
figure(1)
io.imshow(gamma_corrected2)
figure(2)
io.imshow(gamma_corrected3)
figure(3)
io.imshow(gamma_corrected4)
```

Out[95]:

```
<matplotlib.image.AxesImage at 0x21341dcba20>
```

In [98]:

```
from skimage import io
```

```
from skimage.transform import rotate
img = io.imread("C:/Users/Administrator/Desktop/Deep Learning/Tiger.jpg")
img_rotate = rotate(img,30)
io.imshow(img_rotate)
```

Out[98]:

```
<matplotlib.image.AxesImage at 0x21343e659b0>
```

In [100]:

```
#Import libraries
from skimage import io
from skimage import color

#Read image
img = io.imread("C:/Users/Administrator/Desktop/Deep Learning/Tiger.jpg")
#Convert to YPbPr

img_ypbpr= color.rgb2ypbpr(img)
#Convert back to RGB

img_rgb= color.ypbpr2rgb(img_ypbpr)

#Show both figures
figure(0)
io.imshow(img_ypbpr)
figure(1)
io.imshow(img_rgb)
```

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
```

Out[100]:

```
<matplotlib.image.AxesImage at 0x21344514748>
```

In [102]:

```
#Import libraries
from skimage import io
from skimage import color
from pylab import *
```

```
#Read image
img = io.imread("C:/Users/Administrator/Desktop/Deep Learning/Tiger.jpg")


#Convert to YPbPr
img_ypbpr= color.rgb2ypbpr(img)


#Convert back to RGB
img_rgb= color.ypbpr2rgb(img_ypbpr)
io.imsave("Tiger_ypbpr.jpg", img_ypbpr)
```

```
WARNING:root:Lossy conversion from float64 to uint8. Range [-0.3252467450980392, 0.999
9999999999999]. Convert image to uint8 prior to saving to suppress this warning.
```

In [105]:

```
from skimage import io
from skimage.measure import compare_ssim as ssim
img_original = io.imread("C:/Users/Administrator/Desktop/Deep Learning/Tiger.jpg")
img_modified = io.imread('Tiger_ypbpr.jpg')
ssim_original = ssim(img_original, img_original, data_range=img_original.max() - img_o
riginal.min(), multichannel=True)
ssim_different = ssim(img_original, img_modified, data_range=img_modified.max() - img_
modified.min(), multichannel=True)
print(ssim_original,ssim_different)


#SSIM takes three arguments. The first refers to the image;
#the second indicates the range of the pixels (the highest pixel color value less
#the lowest pixel color value). The third argument is multichannel.
#A True value means the image contains more than one channel,
#such as RGB. False means there is only one channel, such as grayscale.
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:5: UserWarning: DEPRE
CATED: skimage.measure.compare_ssim has been moved to skimage.metrics.structural_simil
arity. It will be removed from skimage.measure in version 0.18.
  """
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:6: UserWarning: DEPRE
CATED: skimage.measure.compare_ssim has been moved to skimage.metrics.structural_simil
arity. It will be removed from skimage.measure in version 0.18.

1.0 0.3739397296936154
```

In [116]:

```
#import required packages
import cv2
import numpy as np
```

```
#Read image
image = cv2.imread("C:/Users/Administrator/Desktop/Deep Learning/Tiger.jpg")

#Create a dummy image that stores different contrast and brightness
new_image = np.zeros(image.shape, image.dtype)

#Brightness and contrast parameters
contrast = 3.0
bright = 2

#Change the contrast and brightness
for y in range(image.shape[0]):
    for x in range(image.shape[1]):
        for c in range(image.shape[2]):
            new_image[y,x,c] = np.clip(contrast*image[y,x,c] + bright, 0, 255)
figure(0)
io.imshow(image)
figure(1)
io.imshow(new_image)
```

Out[116]:

```
<matplotlib.image.AxesImage at 0x21344cff9e8>
```

In [126]:

```
#import required packages
import cv2
import numpy as np



#Read image 1
img1 = cv2.imread("C:/Users/Administrator/Desktop/Deep Learning/Tiger.jpg")

#Read image 2
img2 = cv2.imread("C:/Users/Administrator/Desktop/Deep Learning/aircraft/test/000010.j
pg")



#Define alpha and beta
```

```
alpha = 0.30

beta = 0.70


#Blend images

final_image = cv2.addWeighted(img1, alpha, img2, beta, 0.0)


#Show image

io.imshow(final_image)
```

```
---------------------------------------------------------------------------
error                                     Traceback (most recent call last)
<ipython-input-126-b8f2d8b1aad8> in <module>
     16
     17 #Blend images
---> 18 final_image = cv2.addWeighted(img1, alpha, img2, beta, 0.0)
     19
     20 #Show image

error: OpenCV(3.4.1) C:\Miniconda3\conda-bld\opencv-suite_1533128839831\work\modules\c
ore\src\arithm.cpp:659: error: (-209) The operation is neither 'array op array' (where
arrays have the same size and the same number of channels), nor 'array op scalar', nor
'scalar op array' in function cv::arithm_op
```

In [129]:

```
#import required packages

import cv2

import numpy as np


#Read image

image = cv2.imread("C:/Users/Administrator/Desktop/Deep Learning/Tiger.jpg")


#Define font

font  = cv2.FONT_HERSHEY_SIMPLEX


#Write on the image

cv2.putText(image, "I am not a Cat ", (230, 50), font, 0.8, (0, 255, 0), 2, cv2.LINE_A
A)

io.imshow(image)
```

Out[129]:

```
<matplotlib.image.AxesImage at 0x21344debda0>
```

In [131]:

```
#import required packages

import cv2

import numpy as np
```

```
#Read images for different blurring purposes
image_Original = cv2.imread("C:/Users/Administrator/Desktop/Deep Learning/Tiger.jpg")
image_MedianBlur = cv2.imread("C:/Users/Administrator/Desktop/Deep Learning/Tiger.jpg"
)
image_GaussianBlur = cv2.imread("C:/Users/Administrator/Desktop/Deep Learning/Tiger.jp
g")
image_BilateralBlur = cv2.imread("C:/Users/Administrator/Desktop/Deep Learning/Tiger.j
pg")


#Blur images
image_MedianBlur=cv2.medianBlur(image_MedianBlur,9)
image_GaussianBlur=cv2.GaussianBlur(image_GaussianBlur,(9,9),10)
image_BilateralBlur=cv2.bilateralFilter(image_BilateralBlur,9,100,75)


#Show images
figure(0)
io.imshow(image_Original)
figure(1)
io.imshow(image_MedianBlur)
figure(2)
io.imshow(image_GaussianBlur)
figure(3)
io.imshow(image_BilateralBlur)
```

Out[131]:

```
<matplotlib.image.AxesImage at 0x213452b3e10>
```

In [133]:

```
#DILATION CODE :
#Import package
import cv2
#Read image
image = cv2.imread("C:/Users/Administrator/Desktop/Deep Learning/Tiger.jpg")
#Define erosion size
s1 = 0
s2 = 10
```

```
s3 = 10
#Define erosion type
t1 = cv2.MORPH_RECT
t2 = cv2.MORPH_CROSS
t3 = cv2.MORPH_ELLIPSE

#Define and save the erosion template
tmp1 = cv2.getStructuringElement(t1, (2*s1 + 1, 2*s1+1), (s1, s1))
tmp2= cv2.getStructuringElement(t2, (2*s2 + 1, 2*s2+1), (s2, s2))
tmp3 = cv2.getStructuringElement(t3, (2*s3 + 1, 2*s3+1), (s3, s3))

#Apply the erosion template to the image and save in different variables
final1 = cv2.erode(image, tmp1)
final2 = cv2.erode(image, tmp2)
final3 = cv2.erode(image, tmp3)

#Show all the images with different erosions
figure(0)
io.imshow(final1)
figure(1)
io.imshow(final2)
figure(2)
io.imshow(final3)

#EROSION CODE :
#Import packages
import cv2
#Read images
image = cv2.imread("C:/Users/Administrator/Desktop/Deep Learning/Tiger.jpg")

#Define dilation size
d1 = 0
d2 = 10
d3 = 20

#Define dilation type
t1 = cv2.MORPH_RECT
t2 = cv2.MORPH_CROSS
```

```
t3 = cv2.MORPH_ELLIPSE

#Store the dilation templates
tmp1 = cv2.getStructuringElement(t1, (2*d1 + 1, 2*d1+1), (d1, d1))
tmp2 = cv2.getStructuringElement(t2, (2*d2 + 1, 2*d2+1), (d2, d2))
tmp3 = cv2.getStructuringElement(t3, (2*d3 + 1, 2*d3+1), (d3, d3))

#Apply dilation to the images
final1 = cv2.dilate(image, tmp1)
final2 = cv2.dilate(image, tmp2)
final3 = cv2.dilate(image, tmp3)

#Show the images
figure(0)
io.imshow(final1)
figure(1)
io.imshow(final2)
figure(2)
io.imshow(final3)
```

Out[133]:

```
<matplotlib.image.AxesImage at 0x213452304e0>
```

In [141]:

```
#Import packages
import cv2
#Read image
image = cv2.imread("C:/Users/Administrator/Desktop/Deep Learning/Tiger.jpg")
#Define threshold types
"'
0 - Binary
1 - Binary Inverted
2 - Truncated
3 - Threshold To Zero
4 - Threshold To Zero Inverted
"'
```

```
#Apply different thresholds and save in different variables
_, img1 = cv2.threshold(image, 50, 255, 0 )
_, img2 = cv2.threshold(image, 50, 255, 1 )
_, img3 = cv2.threshold(image, 50, 255, 2 )
_, img4 = cv2.threshold(image, 50, 255, 3 )
_, img5 = cv2.threshold(image, 50, 255, 4 )



#Show the different threshold images
figure(0)
io.imshow(img1) #Prints Binary Image
figure(1)
io.imshow(img2) #Prints Binary Inverted Image
figure(2)
io.imshow(img3) #Prints Truncated Image
figure(3)
io.imshow(img4) #Prints Threshold to Zero Image
figure(4)
io.imshow(img5) #Prints Threshold to Zero Inverted Image
```

```
  File "<ipython-input-141-eecc722a2b43>", line 6
    "'
      ^
SyntaxError: EOL while scanning string literal
```

In [145]:

```
#Import packages
import cv2


#Read image
src = cv2.imread("C:/Users/Administrator/Desktop/Deep Learning/Tiger.jpg")


#Apply gaussian blur
cv2.GaussianBlur(src, (3, 3), 0)


#Convert image to grayscale
gray = cv2.cvtColor(src, cv2.COLOR_BGR2GRAY)


#Apply Sobel method to the grayscale image
grad_x = cv2.Sobel(gray, cv2.CV_16S, 1, 0, ksize=3, scale=1, delta=0, borderType=cv2.B
ORDER_DEFAULT) #Horizontal Sobel Derivation
```

```
grad_y = cv2.Sobel(gray, cv2.CV_16S, 0, 1, ksize=3, scale=1, delta=0, borderType=cv2.B
ORDER_DEFAULT) #Vertical Sobel Derivation
abs_grad_x = cv2.convertScaleAbs(grad_x)
abs_grad_y = cv2.convertScaleAbs(grad_y)
grad = cv2.addWeighted(abs_grad_x, 0.5, abs_grad_y, 0.5, 0) #Apply both


#Show the image
io.imshow(grad)#View the image
```

Out[145]:

```
<matplotlib.image.AxesImage at 0x213460f2ef0>
```

In [147]:

```
#Import packages
import cv2
#Read image
src = cv2.imread("C:/Users/Administrator/Desktop/Deep Learning/Tiger.jpg")
#Convert to grayscale
src = cv2.cvtColor(src, cv2.COLOR_BGR2GRAY)
#Apply equalize histogram
src_eqlzd = cv2.equalizeHist(src) #Performs Histogram Equalization
#Show both images
figure(0)
io.imshow(src)
figure(1)
io.imshow(src_eqlzd)
figure(2)
io.imshow(src_eqlzd)
```

Out[147]:

```
<matplotlib.image.AxesImage at 0x21346330e80>
```

In [ ]:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
def extract_sift_features(img):
    sift_initialize = cv2.xfeatures2d.SIFT_create()
```

```
    key_points, descriptors = sift_initialize.detectAndCompute(img, None)
    return key_points, descriptors
def showing_sift_features(img1, img2, key_points):
    return plt.imshow(cv2.drawKeypoints(img1, key_points, img2.copy()))
```

In [5]:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from Sift_Opr import *
print('Make Sure that both the images are in the same folder')
x = input(' Enter the first Image Name')
Image1 = cv2.imread(x)
y = input(' Enter the second Image Name')
Image2 = cv2.imread(y)
Image1_gray = cv2.cvtColor(Image1, cv2.COLOR_BGR2GRAY)
Image2_gray = cv2.cvtColor(Image2, cv2.COLOR_BGR2GRAY)
Image1_key_points, Image1_descriptors = extract_sift_features(Image1_gray)
Image2_key_points, Image2_descriptors = extract_sift_features(Image2_gray)
print( 'Displaying SIFT Features')
showing_sift_features(Image1_gray, Image1, Image1_key_points);
norm = cv2.NORM_L2
bruteForce = cv2.BFMatcher(norm)
matches = bruteForce.match(Image1_descriptors, Image2_descriptors)
matches = sorted(matches, key = lambda match:match.distance)
matched_img = cv2.drawMatches(
    Image1, Image1_key_points,
    Image2, Image2_key_points,
    matches[:100], Image2.copy())
plt.figure(figsize=(186,124))
plt.imshow(matched_img)
```

```
Make Sure that both the images are in the same folder
 Enter the first Image NameTiger
 Enter the second Image NameTiger 1
---------------------------------------------------------------------------
error                                     Traceback (most recent call last)
<ipython-input-5-ef6f4070a999> in <module>
      8 y = input(' Enter the second Image Name')
      9 Image2 = cv2.imread(y)
---> 10 Image1_gray = cv2.cvtColor(Image1, cv2.COLOR_BGR2GRAY)
     11 Image2_gray = cv2.cvtColor(Image2, cv2.COLOR_BGR2GRAY)
     12 Image1_key_points, Image1_descriptors = extract_sift_features(Image1_gray)
```

```
error: OpenCV(3.4.1) C:\Miniconda3\conda-bld\opencv-suite_1533128839831\work\modules\i
mgproc\src\color.cpp:11147: error: (-215) scn == 3 || scn == 4 in function cv::cvtColo
r
```

In [ ]:

```
```

In [4]:

```
import Sift_Operations.py
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-4-fc710d288b49> in <module>
----> 1 import Sift_Operations.py

ModuleNotFoundError: No module named 'Sift_Operations'
```