```
import os
import pandas as pd
```

```
s=pd.read_csv("ham_metadata.csv")
s.head()
```

	lesion_id	image_id	dx	dx_type	age	sex	localization
0	HAM_0000118	ISIC_0027419	bkl	histo	80.0	male	scalp
1	HAM_0000118	ISIC_0025030	bkl	histo	80.0	male	scalp
2	HAM_0002730	ISIC_0026769	bkl	histo	80.0	male	scalp
3	HAM_0002730	ISIC_0025661	bkl	histo	80.0	male	scalp
4	HAM_0001466	ISIC_0031633	bkl	histo	75.0	male	ear

```
d="images for cancer"
from PIL import Image
import numpy as np
l=[]
m=[]
images={}
for path in os.listdir(d):
    full_path = os.path.join(d, path)
    l.append(full_path)
    m.append(path)

d="images for cancer2"
for path in os.listdir(d):
    full_path=os.path.join(d,path)
    l.append(full_path)
    m.append(path)
l=np.array(l)
m=np.array(m)
```

```
for i in range(len(l)):
    images[m[i]]=np.asarray(Image.open(l[i]).resize((299,299)))
```

```
count=0
images1={}
for i in images:
    images1[i[:-4]]=images[i]
    count+=1
print(images[m[0]].shape)
print(len(images1))
```

```
(299, 299, 3)
10015
```

```
s1=[]
for i in range(len(s)):
```

```
for i in range(len(s)):
    img=s["image_id"][i]
    if(img in images1):
        s1.append(images1[img])
    else:
        s1.append("None")
```

```
s["image"]=s1
print(s1.count("None"))
s.head()
print(s.columns)
df=s[s.image!="None"]
df.head()
```

```
C:\Users\shiri\anaconda3\lib\site-packages\ipykernel_launcher.py:2: FutureWarning

0
Index(['lesion_id', 'image_id', 'dx', 'dx_type', 'age', 'sex', 'localization',
       'image'],
      dtype='object')
C:\Users\shiri\anaconda3\lib\site-packages\pandas\core\ops\array_ops.py:57: Futur
  result = libops.scalar_compare(x.ravel(), y, op)
```

	lesion_id	image_id	dx	dx_type	age	sex	localization	image
0	HAM_0000118	ISIC_0027419	bkl	histo	80.0	male	scalp	[[[187, 149, 191], [188, 151, 192], [190, 155,...
1	HAM_0000118	ISIC_0025030	bkl	histo	80.0	male	scalp	[[[25, 13, 22], [25, 14, 22], [26, 13, 23], [2...
2	HAM_0002730	ISIC_0026769	bkl	histo	80.0	male	scalp	[[[186, 128, 136], [186, 127, 135], [190, 131,...
3	HAM_0002730	ISIC_0025661	bkl	histo	80.0	male	scalp	[[[23, 10, 15], [24, 11, 16], [24, 11, 19], [2...
4	HAM_0001466	ISIC_0031633	bkl	histo	75.0	male	ear	[[[123, 82, 104], [129, 86, 109], [133, 91, 11...

```
print(len(df))
k=[]
xtrain=df["image"]
print(len(xtrain))
print(xtrain[0].shape)
```

```
10015
10015
(299, 299, 3)
```

Start coding or generate with AI.

```
import tensorflow as tf
from keras.layers import Conv2D,MaxPooling2D,Flatten,Dense,Dropout
from keras.models import Sequential
```

```
Using TensorFlow backend.
```

```
model = Sequential()
model.add(Conv2D(16,(3,3),input_shape=(75,100,3),padding= "same",activation="relu"))
model.add(Conv2D(16,(3,3),activation="relu"))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(32,(3,3),activation="relu"))
model.add(Conv2D(32,(3,3),activation="relu"))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(64,(3,3),activation="relu"))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(64,(3,3),activation="relu"))
model.add(Flatten(input_shape=(75,100,3)))
model.add(Dense(128,activation="relu"))
model.add(Dropout(0.5))
model.add(Dense(128,activation="relu"))
model.add(Dropout(0.4))
model.add(Dense(128,activation="relu"))
model.add(Dense(62,activation="sigmoid"))
model.add(Dense(7,activation="softmax"))
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_1 (Conv2D)            (None, 75, 100, 16)       448       
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 73, 98, 16)        2320      
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 36, 49, 16)        0         
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 34, 47, 32)        4640      
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 32, 45, 32)        9248      
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 16, 22, 32)        0         
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 14, 20, 64)        18496     
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 7, 10, 64)         0         
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 5, 8, 64)          36928     
_________________________________________________________________
flatten_1 (Flatten)          (None, 2560)              0         
_________________________________________________________________
dense_1 (Dense)              (None, 128)               327808    
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0         
_________________________________________________________________
dense_2 (Dense)              (None, 128)               16512     
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0         
_________________________________________________________________
dense_3 (Dense)              (None, 128)               16512     
_________________________________________________________________
dense_4 (Dense)              (None, 62)                7998      
_________________________________________________________________
```

```
     dense_5 (Dense)              (None, 7)                 441       
     =================================================================
     Total params: 441,351
     Trainable params: 441,351
     Non-trainable params: 0
     _________________________________________________________________
```

```
model.compile(loss="sparse_categorical_crossentropy",optimizer=tf.keras.optimizers.Adam(learning_r
,metrics=["accuracy"])
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator,array_to_img,load_img,img_to_a
datagen=ImageDataGenerator(
    rescale=1/255)
```

```
hist=model.fit_generator(datagen.flow(xtrain,ytrain,batch_size=100),validation_data=(xtest,ytest),
```

```
     Epoch 1/50
     185/185 [==============================] - 404s 2s/step - loss: 1.8795 - accuracy: 0.2124 - va
     Epoch 2/50
     185/185 [==============================] - 200s 1s/step - loss: 1.6420 - accuracy: 0.3314 - va
     Epoch 3/50
     185/185 [==============================] - 201s 1s/step - loss: 1.5402 - accuracy: 0.3738 - va
     Epoch 4/50
     185/185 [==============================] - 200s 1s/step - loss: 1.4540 - accuracy: 0.4196 - va
     Epoch 5/50
     185/185 [==============================] - 197s 1s/step - loss: 1.3976 - accuracy: 0.4471 - va
     Epoch 6/50
     185/185 [==============================] - 206s 1s/step - loss: 1.3140 - accuracy: 0.4845 - va
     Epoch 7/50
     185/185 [==============================] - 205s 1s/step - loss: 1.2619 - accuracy: 0.5082 - va
     Epoch 8/50
     185/185 [==============================] - 208s 1s/step - loss: 1.2225 - accuracy: 0.5224 - va
     Epoch 9/50
     185/185 [==============================] - 208s 1s/step - loss: 1.1824 - accuracy: 0.5436 - va
     Epoch 10/50
     185/185 [==============================] - 207s 1s/step - loss: 1.1395 - accuracy: 0.5606 - va
     Epoch 11/50
     185/185 [==============================] - 204s 1s/step - loss: 1.1488 - accuracy: 0.5591 - va
     Epoch 12/50
     185/185 [==============================] - 206s 1s/step - loss: 1.0847 - accuracy: 0.5813 - va
     Epoch 13/50
     185/185 [==============================] - 206s 1s/step - loss: 1.0543 - accuracy: 0.5953 - va
     Epoch 14/50
     185/185 [==============================] - 207s 1s/step - loss: 1.0268 - accuracy: 0.6018 - va
     Epoch 15/50
     185/185 [==============================] - 209s 1s/step - loss: 1.0074 - accuracy: 0.6108 - va
     Epoch 16/50
     185/185 [==============================] - 207s 1s/step - loss: 0.9903 - accuracy: 0.6186 - va
     Epoch 17/50
     185/185 [==============================] - 208s 1s/step - loss: 0.9516 - accuracy: 0.6318 - va
     Epoch 18/50
     185/185 [==============================] - 209s 1s/step - loss: 0.9344 - accuracy: 0.6407 - va
     Epoch 19/50
     185/185 [==============================] - 208s 1s/step - loss: 0.9257 - accuracy: 0.6448 - va
     Epoch 20/50
     185/185 [==============================] - 208s 1s/step - loss: 0.9073 - accuracy: 0.6547 - va
     Epoch 21/50
     185/185 [==============================] - 153s 828ms/step - loss: 0.8840 - accuracy: 0.6621 -
     Epoch 22/50
     185/185 [==============================] - 164s 888ms/step - loss: 0.8617 - accuracy: 0.6695 -
     Epoch 23/50
     185/185 [==============================] - 173s 933ms/step - loss: 0.8415 - accuracy: 0.6755 -
```

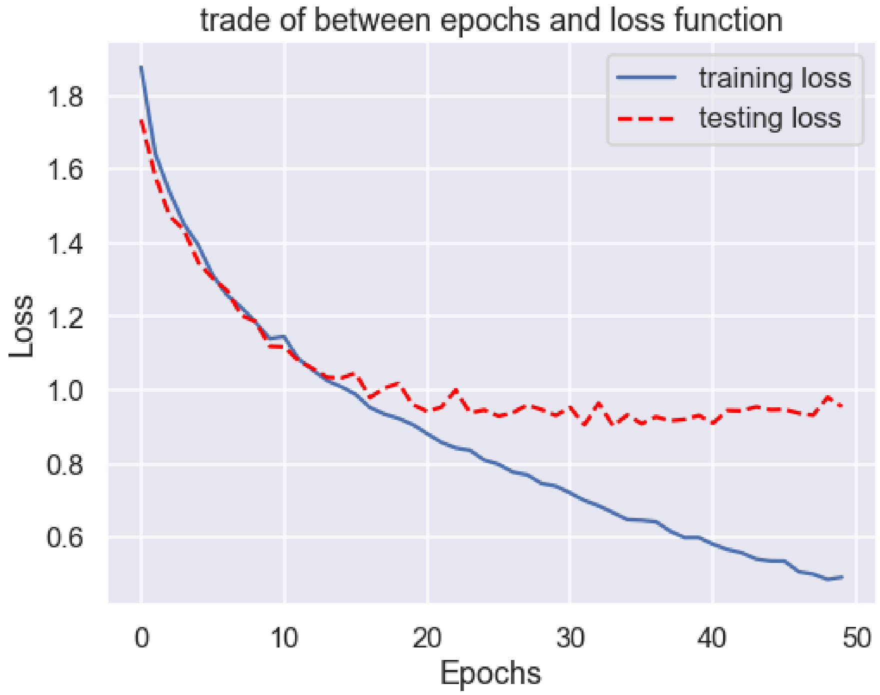
```
    Epoch 24/50
    185/185 [==============================] - 182s 982ms/step - loss: 0.8345 - accuracy: 0.6780 -
    Epoch 25/50
    185/185 [==============================] - 174s 940ms/step - loss: 0.8083 - accuracy: 0.6894 -
    Epoch 26/50
    185/185 [==============================] - 169s 915ms/step - loss: 0.7981 - accuracy: 0.6933 -
    Epoch 27/50
    185/185 [==============================] - 170s 917ms/step - loss: 0.7789 - accuracy: 0.7046 -
    Epoch 28/50
    185/185 [==============================] - 174s 942ms/step - loss: 0.7672 - accuracy: 0.7030 -
    Epoch 29/50
    185/185 [==============================] - 168s 908ms/step - loss: 0.7440 - accuracy: 0.7144 -
```

```
hist1=model.evaluate(xtest,ytest)
```

```
    5287/5287 [==============================] - 11s 2ms/step
```

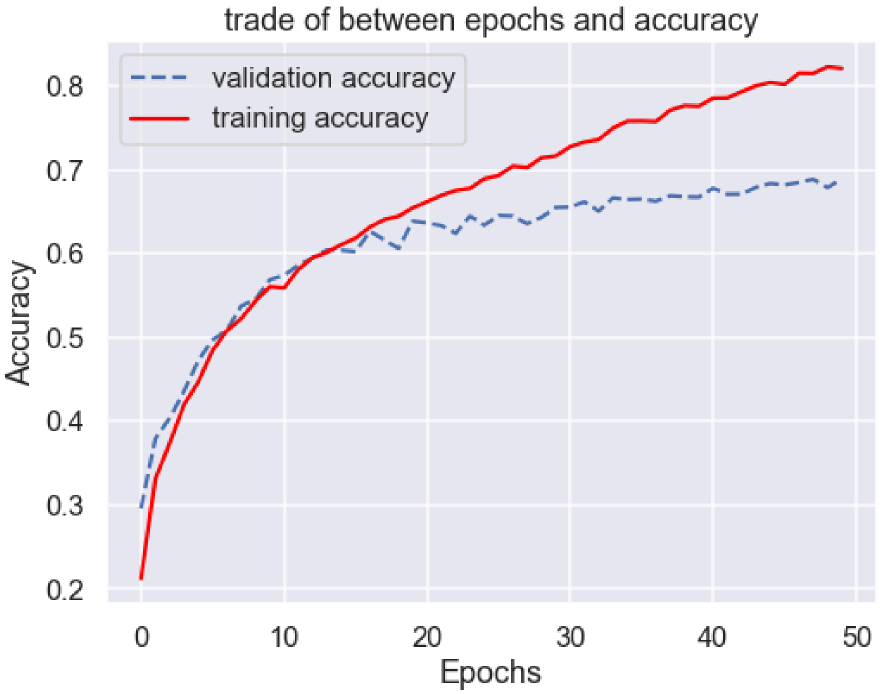
```
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_context('talk')
plt.figure(figsize=(8,6))
plt.title("trade of between epochs and loss function")
#plt.plot(range(0,5),hist.history["loss"])
plt.plot(range(0,50),hist.history["loss"],label="training loss")
plt.plot(range(0,50),hist.history["val_loss"],linestyle="--",label="testing loss",color="red")
plt.xlabel("Epochs")
plt.legend()
plt.ylabel("Loss")
```

```
    Text(0, 0.5, 'Loss')
```



```
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_context('talk')
plt.figure(figsize=(8,6))
plt.title("trade of between epochs and accuracy")
#plt.plot(range(0,5),hist.history["accuracy"])
plt.plot(range(0,50),hist.history["val_accuracy"],linestyle="--",label="validation accuracy")
plt.plot(range(0,50),hist.history["accuracy"],label="training accuracy",color="red")
plt.xlabel("Epochs")
plt.legend()
plt.ylabel("Accuracy")
```

```
    Text(0, 0.5, 'Accuracy')
```



```
d1= {
    'nv': 'Melanocytic nevi',
    'mel': 'Melanoma',
    'bkl': 'Benign keratosis-like lesions ',
    'bcc': 'Basal cell carcinoma',
    'akiec': 'Actinic keratoses',
    'vasc': 'Vascular lesions',
    'df': 'Dermatofibroma'
}

d2= {
    'nv':0,
    'mel':1,
    'bkl':2,
    'bcc':3,
    'akiec':4,
    'vasc':5,
    'df':6
}
```

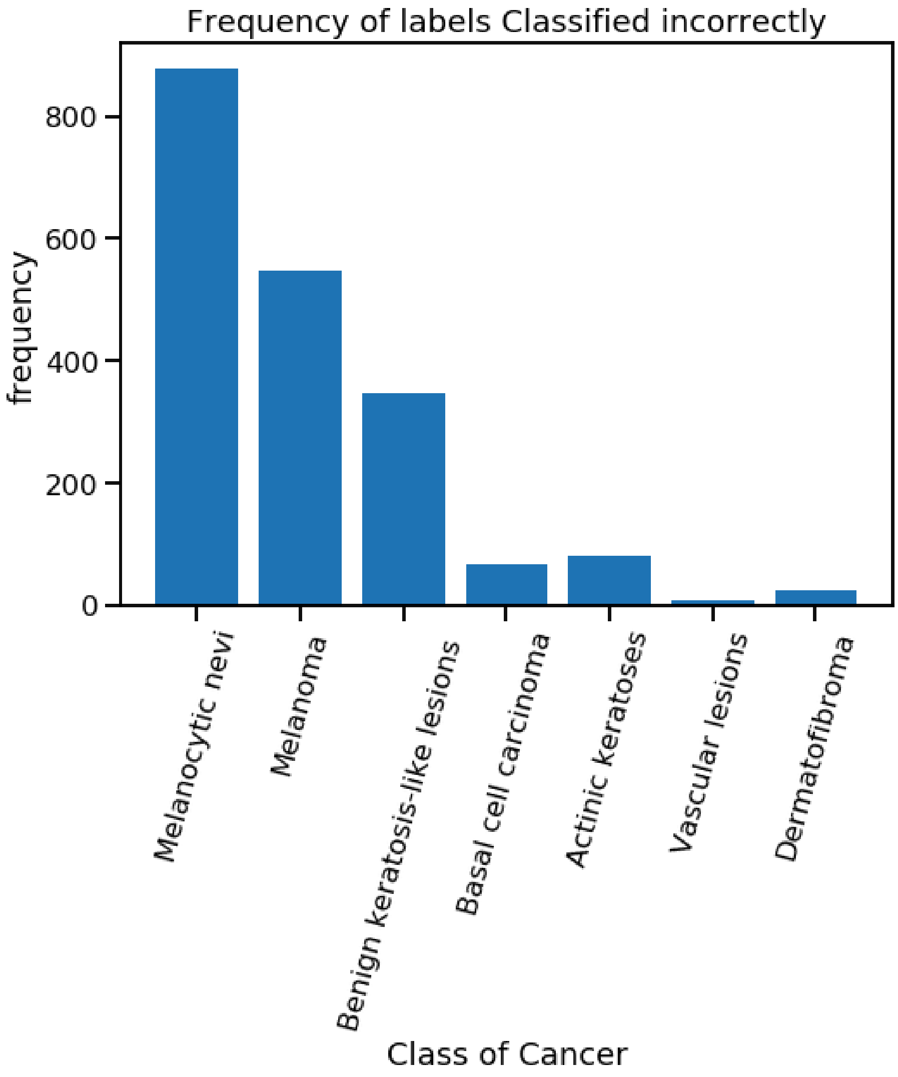
```
label=[];type=[]
for i in range(len(df)):
    label.append(d2[df["dx"][i]])
    type.append(d1[df["dx"][i]])
df["label"]=label
df["type"]=type
```

```
df.head()
```

	lesion_id	image_id	dx	dx_type	age	sex	localization	ima
0	HAM_0000118	ISIC_0027419	bkl	histo	80.0	male	scalp	[[[190, 153, 194], [192, 154, 196], [191, 153
1	HAM_0000118	ISIC_0025030	bkl	histo	80.0	male	scalp	[[[23, 13, 22], [24, 14, 24], [25, 14, 28], [
2	HAM_0002730	ISIC_0026769	bkl	histo	80.0	male	scalp	[[[185, 127, 137], [189, 133, 147], [194, 136
3	HAM_0002730	ISIC_0025661	bkl	histo	80.0	male	scalp	[[[24, 11, 17], [26, 13, 22], [38, 21, 32], [
4	HAM_0001466	ISIC_0031633	bkl	histo	75.0	male	ear	[[[134, 90, 113], [147, 102, 125], [159, 115

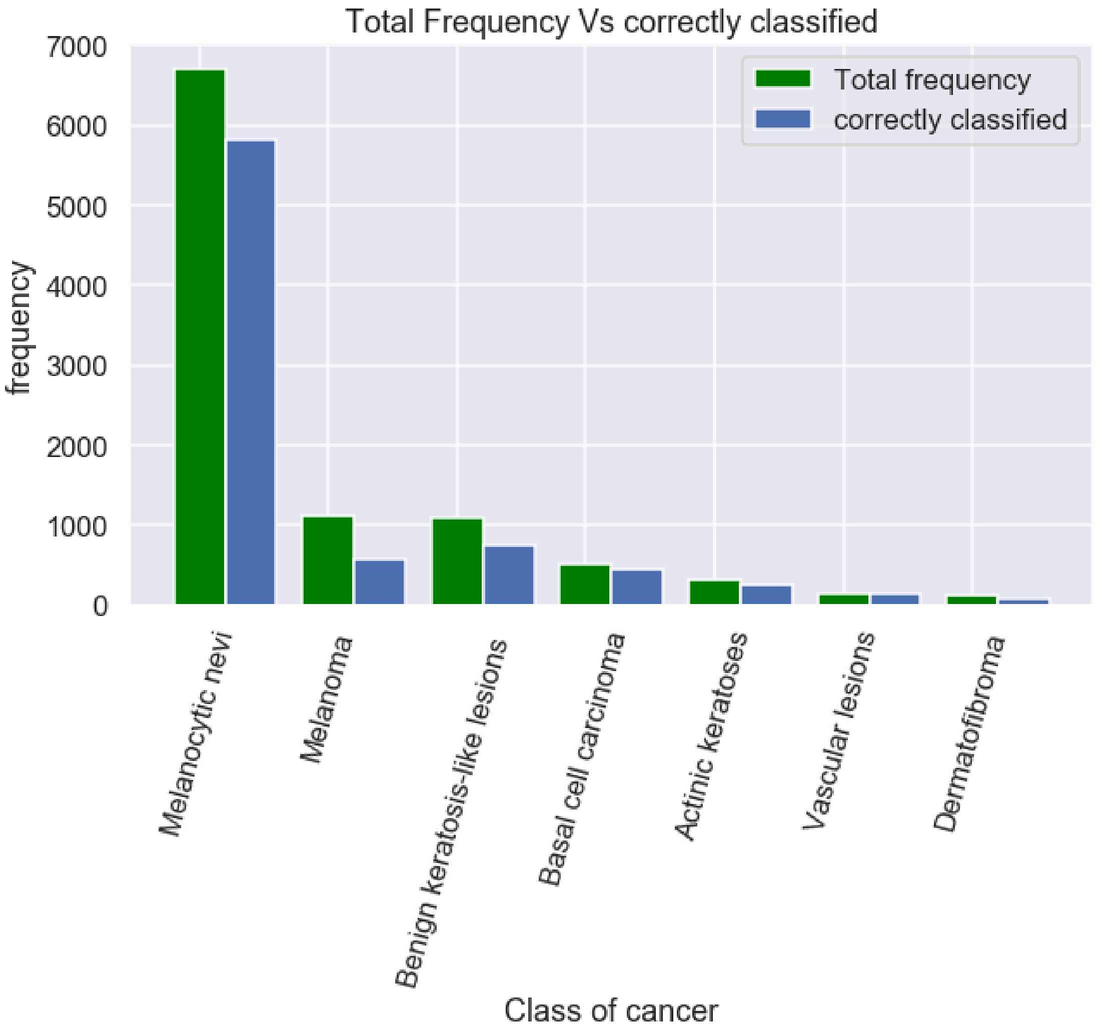
```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(label,ypred)
acc=0
for i in range(len(ypred)):
    acc+=(ypred[i]==label[i])
print(acc)
print(cm)
cma=cm
cm=pd.DataFrame(cm)
emp=[]
emp2=[]
actual=[]
cm.index=x.index
cm.columns=x.index
for i in range(len(cma)):
    empc=0
    empw=0
    for j in range(len(cma)):
        if(i!=j):
            empc+=cma[i][j]
        else:
            empw+=cma[i][j]
    emp.append(empc)
    emp2.append(empw)
    actual.append(sum(cma[i]))
print(actual)

print(emp)
print(emp2)
plt.figure(figsize=(8,6))
plt.title("Frequency of labels Classified incorrectly")
plt.xlabel("Class of Cancer")
plt.ylabel("frequency")
plt.bar(x.index,emp)
plt.xticks(rotation="75")
```

```
8063
[[5828  214  415  156   38   34   20]
 [ 284  566  214   22   24    1    2]
 [ 226   58  753   39   20    1    2]
 [  17    1   19  446   28    2    1]
 [   0    6   33   42  246    0    0]
 [   3    1    1    1    0  134    2]
 [  13    0    5    5    2    0   90]]
[6705, 1113, 1099, 514, 327, 142, 115]
[877, 547, 346, 68, 81, 8, 25]
[5828, 566, 753, 446, 246, 134, 90]
([0, 1, 2, 3, 4, 5, 6], <a list of 7 Text xticklabel objects>)
```



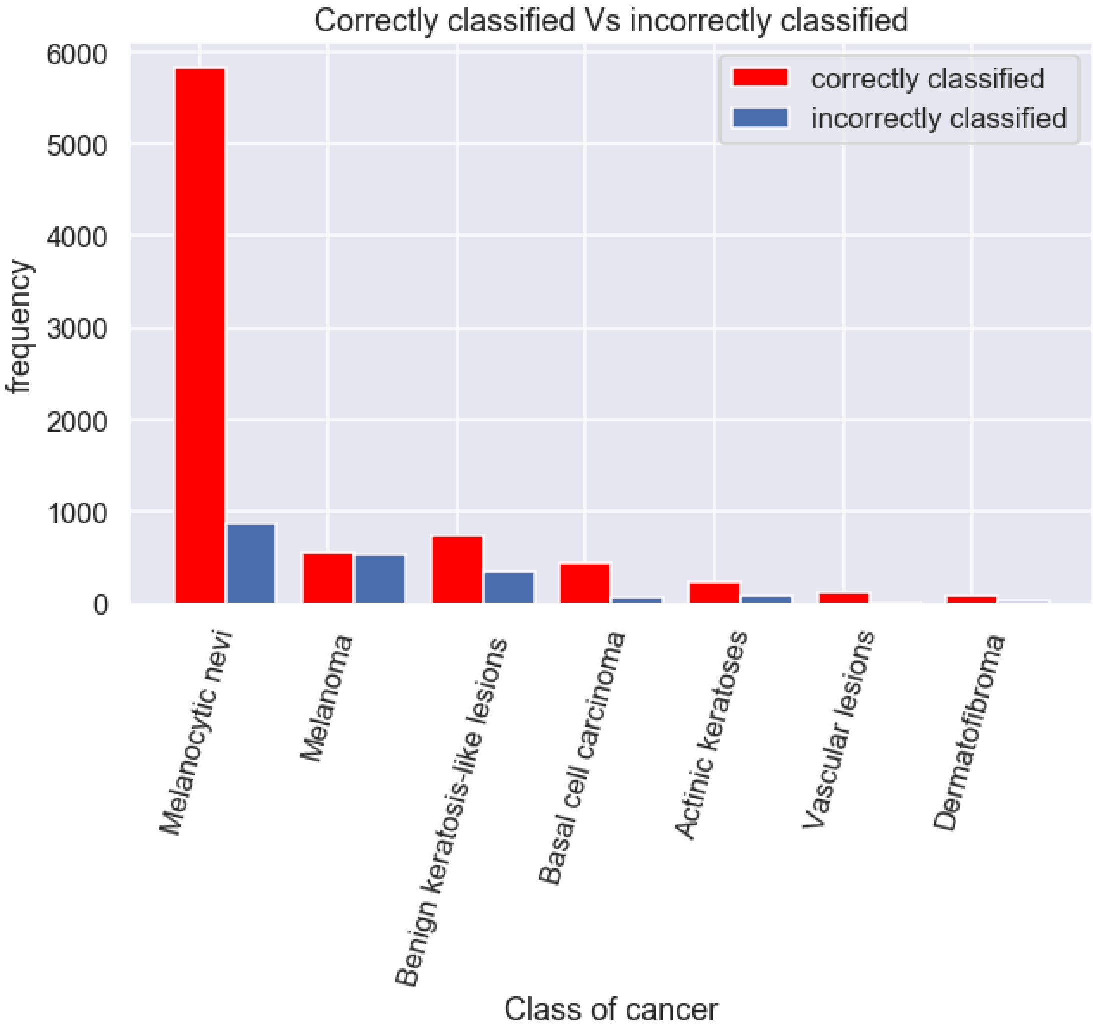
```
plt.figure(figsize=(10,6))
a=np.arange(7)
plt.title("Total Frequency Vs correctly classified")
plt.xlabel("Class of cancer")
plt.ylabel("frequency")
bar_width=0.4
print(a)
plt.bar(x.index,actual,width=bar_width,label="Total frequency",color="green")
plt.xticks(rotation="75")
plt.bar(a+bar_width,emp2,width=bar_width,label="correctly classified")
plt.legend()
plt.show()
```

```
[0 1 2 3 4 5 6]
```



```
plt.figure(figsize=(10,6))
a=np.arange(7)
plt.title("Correctly classified Vs incorrectly classified")
plt.xlabel("Class of cancer")
plt.ylabel("frequency")
bar_width=0.4
print(a)
plt.bar(x.index,emp2,width=bar_width,label="correctly classified",color="red")
plt.xticks(rotation="75")
plt.bar(a+bar_width,emp,width=bar_width,label="incorrectly classified")
plt.legend()
plt.show()
lb=x.index
```

```
    [0 1 2 3 4 5 6]
```



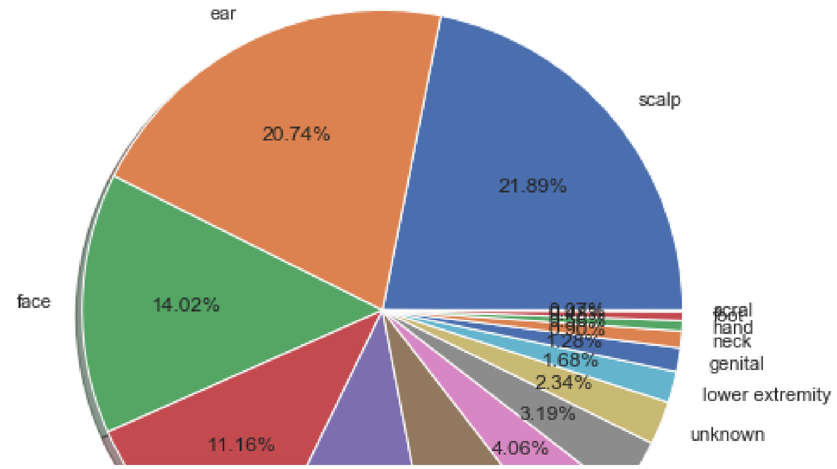
```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(ytest,ypred1)
acc=0
print(cm)
cm=pd.DataFrame(cm)
cm.index=x.index
cm.columns=x.index
```

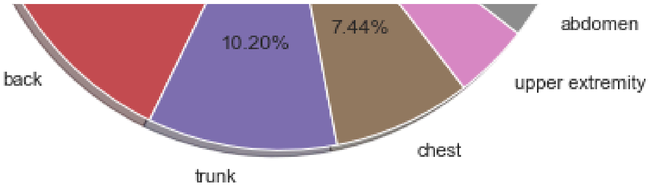
```
    [[623  24  41  16   6   5   4]
     [ 99 429  90  38  62   5  15]
     [ 80 121 349  48  89   6  18]
```

```
      [ 13  22  39 401 111  22  31]
      [  1  51  67 148 378   0  28]
      [  0   2   1  10   2 512   3]
      [  7  12  15  41  32   5 480]]
```

```
import matplotlib.pyplot as plt
vals=df["localization"].unique()
print(vals)
cnts=df["localization"].value_counts()
plt.figure(figsize=(8,8))
plt.pie(cnts,labels=vals,autopct="%1.2f%%",shadow=True,pctdistance=0.65)
```

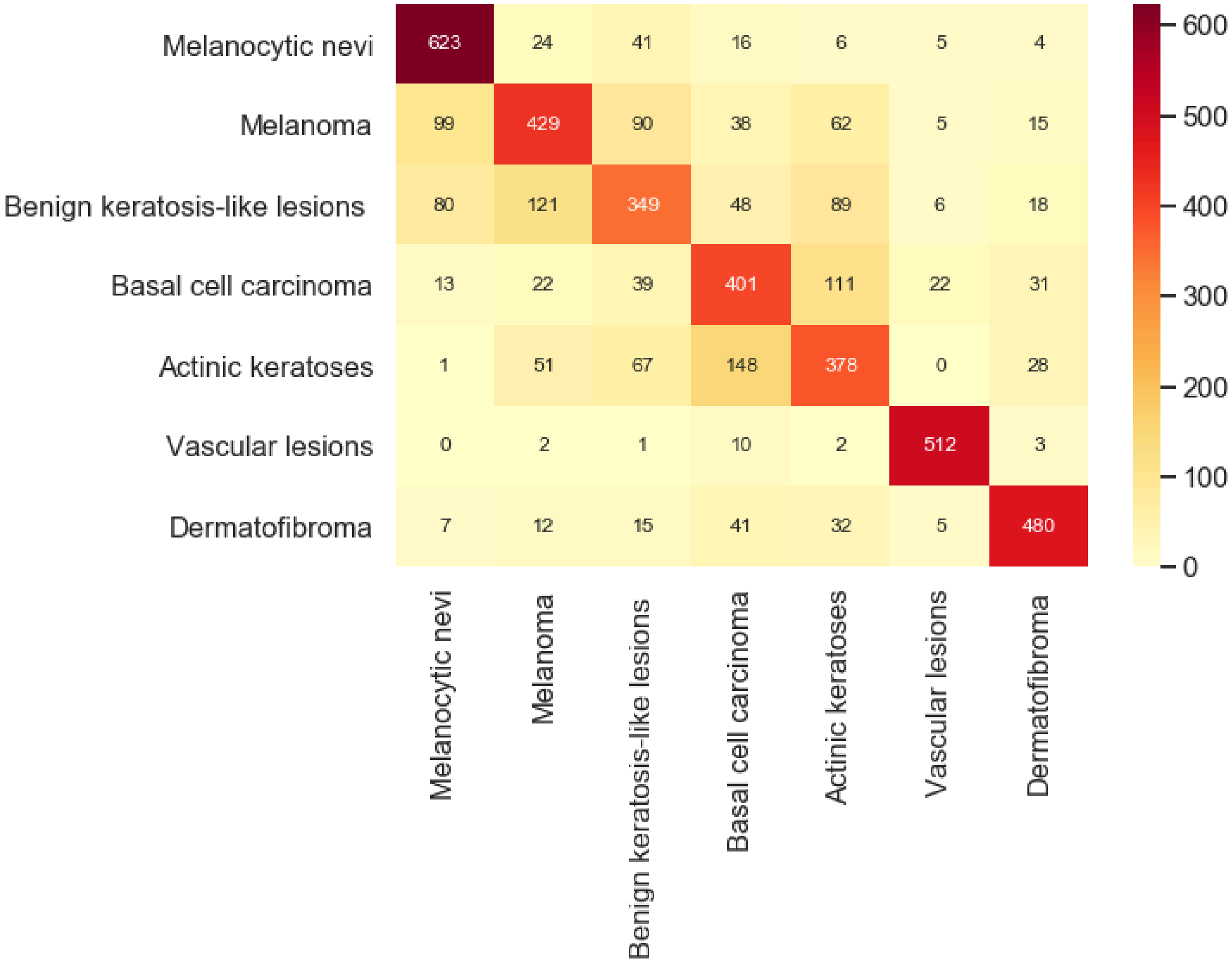
```
['scalp' 'ear' 'face' 'back' 'trunk' 'chest' 'upper extremity' 'abdomen'
 'unknown' 'lower extremity' 'genital' 'neck' 'hand' 'foot' 'acral']
([<matplotlib.patches.Wedge at 0x2d0dc77b708>,
  <matplotlib.patches.Wedge at 0x2d0dc783208>,
  <matplotlib.patches.Wedge at 0x2d0dc783e88>,
  <matplotlib.patches.Wedge at 0x2d0dc78cbc8>,
  <matplotlib.patches.Wedge at 0x2d0dc797b48>,
  <matplotlib.patches.Wedge at 0x2d0dc7a1848>,
  <matplotlib.patches.Wedge at 0x2d0dc7aa288>,
  <matplotlib.patches.Wedge at 0x2d0dc7aaf08>,
  <matplotlib.patches.Wedge at 0x2d0dc797888>,
  <matplotlib.patches.Wedge at 0x2d0dc7a1588>,
  <matplotlib.patches.Wedge at 0x2d0dc744e48>,
  <matplotlib.patches.Wedge at 0x2d0dc7cf0c8>,
  <matplotlib.patches.Wedge at 0x2d0dc7cfd48>,
  <matplotlib.patches.Wedge at 0x2d0dc7d8a08>,
  <matplotlib.patches.Wedge at 0x2d0dc7e26c8>],
 [Text(0.850044637801859, 0.6981576567970206, 'scalp'),
  Text(-0.48434367590701677, 0.9876290819983374, 'ear'),
  Text(-1.0997116076210351, 0.02518690261938371, 'face'),
  Text(-0.7910606058887917, -0.7643448945409774, 'back'),
  Text(-0.14399207613610546, -1.0905348605202927, 'trunk'),
  Text(0.4515570372814075, -1.0030434896262663, 'chest'),
  Text(0.7770241418489049, -0.7786099684591594, 'upper extremity'),
  Text(0.9327513585770874, -0.5830736686496809, 'abdomen'),
  Text(1.0193916346283005, -0.413328797992399, 'unknown'),
  Text(1.0632811366512243, -0.2818390044718449, 'lower extremity'),
  Text(1.0848325959235923, -0.1820391134390633, 'genital'),
  Text(1.0947359222518052, -0.10748609459595235, 'neck'),
  Text(1.0985090016252645, -0.057253588082013494, 'hand'),
  Text(1.0997919697781062, -0.021392129664751052, 'foot'),
  Text(1.099997348309977, -0.0024153076448207023, 'acral')],
 [Text(0.5022991041556439, 0.4125477062891485, '21.89%'),
  Text(-0.28620308121778265, 0.5835990029990176, '20.74%'),
  Text(-0.6498295863215208, 0.014883169729635829, '14.02%'),
  Text(-0.46744490347974055, -0.4516583467742139, '11.16%'),
  Text(-0.08508622680769867, -0.6444069630347184, '10.20%'),
  Text(0.266829158393559, -0.5927075165973391, '7.44%'),
  Text(0.45915062927435285, -0.46008770863495785, '4.06%'),
  Text(0.5511712573410061, -0.3445435314748114, '3.19%'),
  Text(0.6023677840985412, -0.24423974426823578, '2.34%'),
  Text(0.6283024898393598, -0.16654122991518108, '1.68%'),
  Text(0.641037443045759, -0.10756856703217375, '1.28%'),
  Text(0.6468894086033393, -0.06351451044306275, '0.90%'),
  Text(0.6491189555058381, -0.03383166568482615, '0.56%'),
  Text(0.6498770730506991, -0.01264080389280744, '0.48%'),
  Text(0.6499984330922591, -0.0014272272446667787, '0.07%')])
```




10.20%
7.44%
abdomen
back
upper extremity
chest
trunk

```
import seaborn as sn
plt.figure(figsize=(9,6))
sn.heatmap(cm, annot=True, annot_kws={"size": 12},cmap= 'YlOrRd',fmt="g")
```

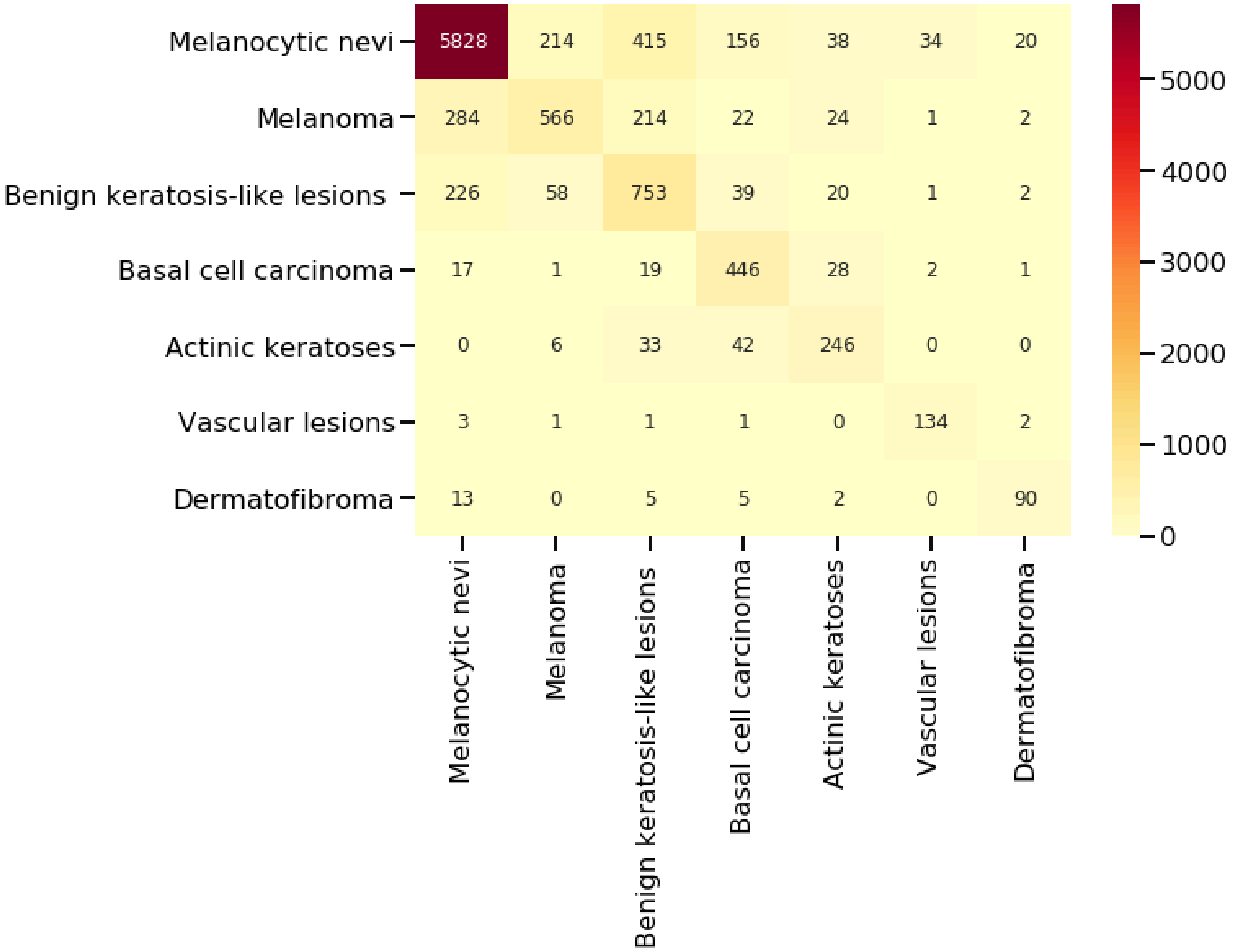
```
<matplotlib.axes._subplots.AxesSubplot at 0x279d27c9b08>
```



	Melanocytic nevi	Melanoma	Benign keratosis-like lesions	Basal cell carcinoma	Actinic keratoses	Vascular lesions	Dermatofibroma
Melanocytic nevi	623	24	41	16	6	5	4
Melanoma	99	429	90	38	62	5	15
Benign keratosis-like lesions	80	121	349	48	89	6	18
Basal cell carcinoma	13	22	39	401	111	22	31
Actinic keratoses	1	51	67	148	378	0	28
Vascular lesions	0	2	1	10	2	512	3
Dermatofibroma	7	12	15	41	32	5	480

```
import seaborn as sn
plt.figure(figsize=(9,6))
sn.heatmap(cm, annot=True, annot_kws={"size": 12},cmap= 'YlOrRd',fmt="g")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x279d4346c48>
```



```
ypred1=[]
for i in range(len(xtest)):
    ypred1.append(np.argmax(model.predict(xtest[i].reshape(1,75,100,3))))

#print(np.argmax(model.predict(xtest[1].reshape(1,75,100,3))))


s1=np.array(s1)


from sklearn import metrics
f_sco=metrics.precision_recall_fscore_support(label,ypred,average="weighted")
print(f_sco)0.8186782169714542
```

```
(0.8186782169714542, 0.8050923614578133, 0.807932103655772, None)
```

```
from sklearn import metrics
f_sco=metrics.precision_recall_fscore_support(ytest,ypred1,average="weighted")
print(f_sco)
```

```
(0.6854138384138033, 0.6892655367231638, 0.6856373865227726, None)
```

```
ypred=[]
for i in range(len(s1)):
    ypred.append(np.argmax(model.predict(s1[i].reshape(1,75,100,3))))
    if(i%1000==0):
        print(i)
```

```
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
```

```
s1=np.array(s1)
```

```
s1=s1/255
```

```
s1=np.array(s1)
sdup=s1
ldup=label
print(s1.shape)
```

```
(10015, 75, 100, 3)
```

```
from sklearn.model_selection import train_test_split
```

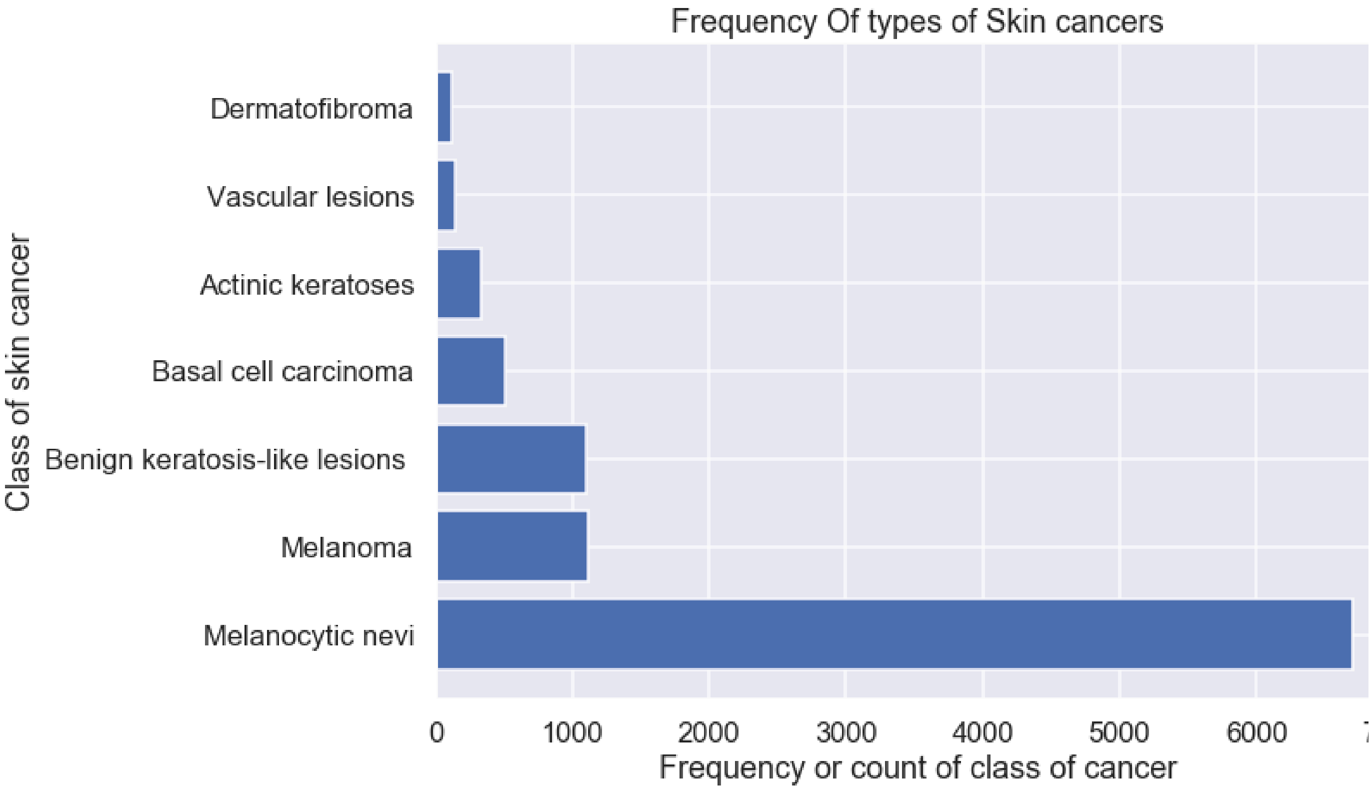
```
xtrain,xtest,ytrain,ytest=train_test_split(inp,oup,train_size=0.8)
```

```
xtest=xtest/255
```

```
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
```

```
lbl=df["type"]
bars=["blue","red","green","yellow","indigo","pink","grey"]
x=pd.DataFrame(lbl.value_counts())
print(x)
import seaborn as sns
sns.set_context('talk')
ax=plt.figure(figsize=(10,7))
plt.title("Frequency Of types of Skin cancers")
plt.barh(x.index,x["type"])
plt.ylabel("Class of skin cancer")
#plt.legend(b,list(x.index))
plt.xlabel("Frequency or count of class of cancer")
```

```
                                  type
Melanocytic nevi                  6705
Melanoma                          1113
Benign keratosis-like lesions     1099
Basal cell carcinoma               514
Actinic keratoses                  327
Vascular lesions                   142
Dermatofibroma                     115
Text(0.5, 0, 'Frequency or count of class of cancer')
```



```
li=[]
for i in range(len(x.index)):
    te=df[df.type==x.index[i]]
    valu=te["sex"].value_counts()
    valu=list(valu)
    if(len(valu)==2):
        valu.append(0)
    li.append(valu)
print(li)
```

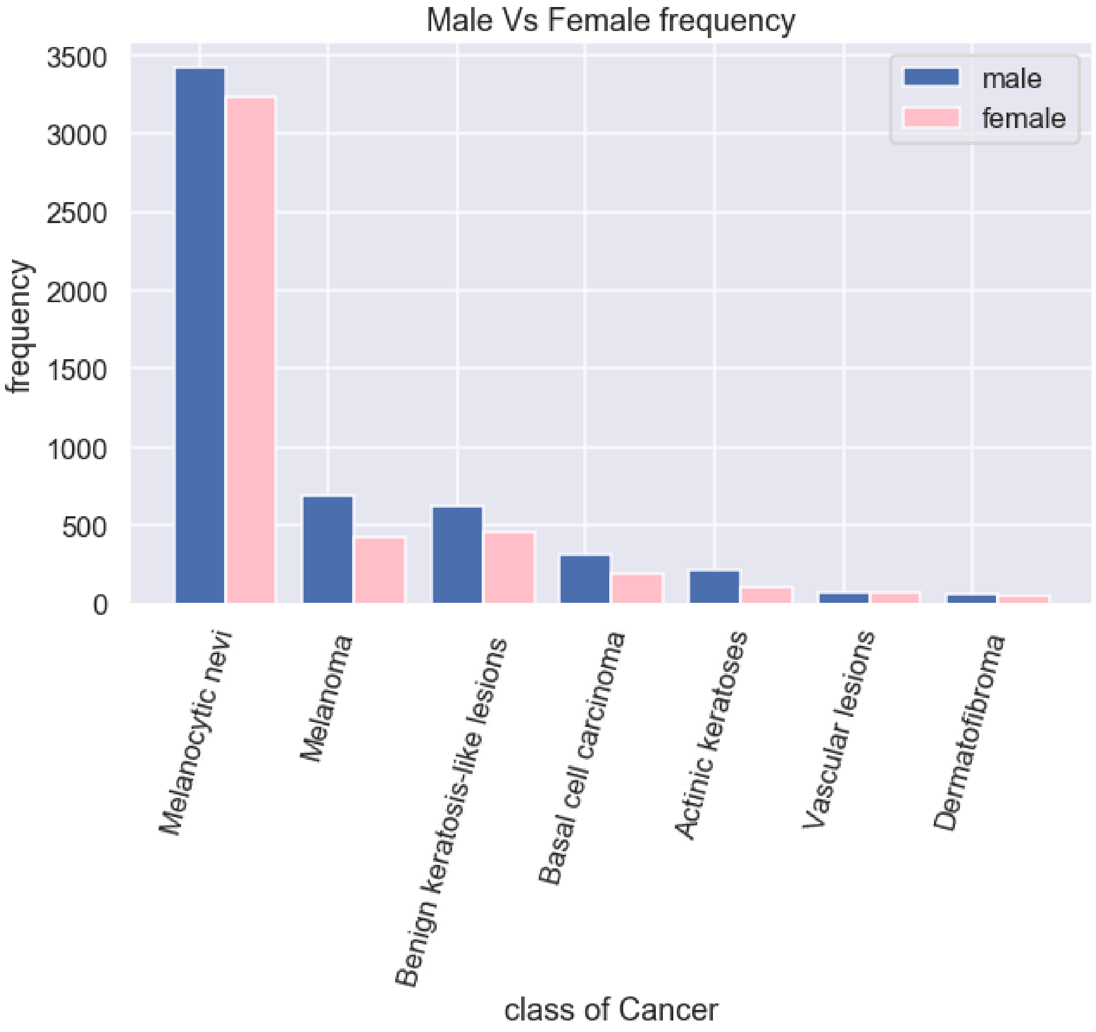
```
    [[3421, 3237, 47], [689, 424, 0], [626, 463, 10], [317, 197, 0], [221, 106, 0], [73, 69, 0], [
```

```
male=[];female=[];unknown=[]
for i in range(len(li)):
    male.append(li[i][0])
    female.append(li[i][1])
    unknown.append(li[i][2])

print(male)
print(female)
```

```
    [3421, 689, 626, 317, 221, 73, 63]
    [3237, 424, 463, 197, 106, 69, 52]
```

```
plt.figure(figsize=(10,6))
plt.title("Male Vs Female frequency")
plt.xlabel("class of Cancer")
plt.ylabel("frequency")
a=np.arange(7)
bar_width=0.4
print(a)

plt.bar(x.index,male,width=bar_width,label="male")
plt.xticks(rotation="75")
plt.bar(a+bar_width,female,width=bar_width,label="female",color="pink")
plt.legend()
plt.show()

lb=x.index
```

```
[0 1 2 3 4 5 6]
```



```
from tensorflow.keras.preprocessing.image import ImageDataGenerator,array_to_img,load_img,img_to_a
datagen1=ImageDataGenerator(
    rotation_range=90,
    width_shift_range=0.2,
    height_shift_range=0.2,
    brightness_range=None,
    shear_range=0.1,
    zoom_range=0.1,
)


derimages=[]
for i in range(len(df)):
    if(df["type"][i]=="Dermatofibroma"):
        array=df["image"][i].reshape((1,)+df["image"][i].shape)
        derimages.append(df["image"][i])
```

```
for i in range(len(df)):
    if(df["type"][i]=="Dermatofibroma"):
        array=df["image"][i].reshape((1,)+df["image"][i].shape)
        derimages.append(df["image"][i])
        j=0
        for i in datagen1.flow(array,save_to_dir="new",save_prefix="der",save_format="jpeg"):
            j+=1
            if(j>10):
                break
```

```
     KeyboardInterrupt
```

```
vasimages=[]
for i in range(len(df)):
    if(df["type"][i]=="Vascular lesions"):
        vasimages.append(df["image"][i])
```

```
for i in range(len(df)):
    if(df["type"][i]=="Vascular lesions"):
        array=df["image"][i].reshape((1,)+df["image"][i].shape)
        j=0
        for i in datagen1.flow(array,save_to_dir="new1",save_prefix="vas",save_format="jpeg"):
            j+=1
            if(j>20):
                break
```

```
actimages=[]
for i in range(len(df)):
    if(df["type"][i]=="Actinic keratoses"):
        actimages.append(df["image"][i])
```

```
for i in range(len(df)):
    if(df["type"][i]=="Actinic keratoses"):
        array=df["image"][i].reshape((1,)+df["image"][i].shape)
        j=0
        for i in datagen1.flow(array,save_to_dir="new2",save_prefix="act",save_format="jpeg"):
            j+=1
            if(j>10):
                break
```

```
basimages=[]
for i in range(len(df)):
    if(df["type"][i]=="Basal cell carcinoma"):
        basimages.append(df["image"][i])
```

```
for i in range(len(df)):
    if(df["type"][i]=="Basal cell carcinoma"):
        array=df["image"][i].reshape((1,)+df["image"][i].shape)
        j=0
        for i in datagen1.flow(array,save_to_dir="new3",save_prefix="bas",save_format="jpeg"):
            j+=1
            if(j>5):
                break
```

```
benimages=[]
for i in range(len(df)):
    if(df["type"][i]=="Benign keratosis-like lesions "):
        benimages.append(df["image"][i])
```

```
for i in range(len(df)):
    if(df["type"][i]=="Benign keratosis-like lesions "):
        array=df["image"][i].reshape((1,)+df["image"][i].shape)
        j=0
        for i in datagen1.flow(array,save_to_dir="new4",save_prefix="ben",save_format="jpeg"):
            j+=1
            if(j>1):
                break
```

```
melimages=[]
for i in range(len(df)):
    if(df["type"][i]=="Melanoma"):
        melimages.append(df["image"][i])
```

```
for i in range(len(df)):
    if(df["type"][i]=="Melanoma"):
        array=df["image"][i].reshape((1,)+df["image"][i].shape)
        j=0
        for i in datagen1.flow(array,save_to_dir="new5",save_prefix="mel",save_format="jpeg"):
            j+=1
            if(j>3):
                break
```

```
d="new1"
vasm=[]
vas=[]
for path in os.listdir(d):
    full_path = os.path.join(d, path)
    vas.append(full_path)
    vasm.append(path)
```

```
print(x)
```

```
                                type
Melanocytic nevi                6705
Melanoma                        1113
Benign keratosis-like lesions   1099
Basal cell carcinoma             514
Actinic keratoses                327
Vascular lesions                 142
```

```
Dermatofibroma                    115
```

```
for i in range(len(vas)):
    vasimages.append(np.asarray(Image.open(vas[i])))
```

```
d="new2"
actm=[]
act=[]
for path in os.listdir(d):
    full_path = os.path.join(d, path)
    act.append(full_path)
    actm.append(path)
```

```
for i in range(len(act)):
    actimages.append(np.asarray(Image.open(act[i])))
```

```
d="new3"
bas=[]
basm=[]
for path in os.listdir(d):
    full_path = os.path.join(d, path)
    bas.append(full_path)
    basm.append(path)
for i in range(len(bas)):
    basimages.append(np.asarray(Image.open(bas[i])))
```

```
d="new"
der=[]
derm=[]
for path in os.listdir(d):
    full_path = os.path.join(d, path)
    der.append(full_path)
    derm.append(path)
print(len(derm))

for i in range(len(der)):
    derimages.append(np.asarray(Image.open(der[i])))
```

```
3033
```

```
d="new4"
ben=[]
benm=[]
for path in os.listdir(d):
    full_path = os.path.join(d, path)
    ben.append(full_path)
    benm.append(path)
for i in range(2300):
    index=randint(0,len(ben)-1)
    benimages.append(np.asarray(Image.open(ben[index])))
```

```
import random
from random import randint
```

```
d="new5"
mel=[]
melm=[]
for path in os.listdir(d):
    full_path = os.path.join(d, path)
    mel.append(full_path)
    melm.append(path)
for i in range(2500):
    index=randint(0,len(mel)-1)
    melimages.append(np.asarray(Image.open(mel[index])))
```

```
melt=[]
for i in range(len(df)):
    if(df["type"][i]=="Melanocytic nevi"):
        melt.append(df["image"][i])
print(len(melt))
```

```
     6705
```

```
import random
from random import randint
```

```
melt1=[]
for i in range(3600):
    index=randint(0,len(melt)-1)
    melt1.append(melt[index])
```

```
melt=melt1
```

```
v=[]
v.extend(melt)
print(len(v))
```

```
     3600
```

```
v.extend(melimages)
```

```
v.extend(benimages)
v.extend(basimages)
v.extend(actimages)
v.extend(vasimages)
v.extend(derimages)
```

```
print(len(v))
```

```
     22993
```

```
print(len(benimages),len(basimages),len(actimages),len(vasimages),len(derimages))
print(len(melimages))
```

```
    3399 3161 3338 2734 3148
    3613
```

```
lb6=[0 for i in range(len(melt))]
print(len(lb6))
```

```
    3600
```

```
lb5=[1 for i in range(len(melimages))]
```

```
lb4=[2 for i in range(len(benimages))]
lb3=[3 for i in range(len(basimages))]
lb2=[4 for i in range(len(actimages))]
lbl1=[5 for i in range(len(vasimages))]
lbl0=[6 for i in range(len(derimages))]
```

```
out=[]
out.extend(lb6)
out.extend(lb5)
out.extend(lb4)
out.extend(lb3)
out.extend(lb2)
out.extend(lbl1)
out.extend(lbl0)
```

```
print(len(out))
```

```
    22993
```

```
print(len(v))
```

```
    22993
```

```
inp=[]
oup=[]
import random
from random import randint
for i in range(len(v)):
    inp.append(v[i])
    oup.append(out[i])
```

```
for i in range(len(v)):
    if(v[i]=='new2\\Actinic keratoses_0_4111.jpeg'):
        print(i)
```

```
    C:\Users\shiri\anaconda3\lib\site-packages\ipykernel_launcher.py:2: FutureWarning: elementwise
```

```
inp=np.array(inp)
```

```
print(inp.shape)
```

```
    (22993, 75, 100, 3)
```

```
inp=inp.reshape(22993,22500)
```

```
75*100*3
import matplotlib.pyplot as plt
```

```
    10015
```

```
import numpy
import pygad
import pygad.nn
import pygad.gann
```

Start coding or generate with AI.

```
def fitness_func(solution, sol_idx):
    global GANN_instance, data_inputs, data_outputs

    predictions = pygad.nn.predict(last_layer=GANN_instance.population_networks[sol_idx],
                                   data_inputs=data_inputs)
    correct_predictions = numpy.where(predictions == data_outputs)[0].size
    solution_fitness = (correct_predictions/data_outputs.size)*100

    return solution_fitness
```

```
def callback_generation(ga_instance):
    global GANN_instance

    population_matrices = pygad.gann.population_as_matrices(population_networks=GANN_instance.popu
                                                            population_vectors=ga_instance.populat

    GANN_instance.update_population_trained_weights(population_trained_weights=population_matrices

    print("Generation = {generation}".format(generation=ga_instance.generations_completed))
    print("Accuracy   = {fitness}".format(fitness=ga_instance.best_solution()[1]))
```