In [2]:
```
pip install tensorflow
```

```
Requirement already satisfied: tensorflow in c:\programdata\anaconda3\lib\sit
e-packages (2.9.1)
Requirement already satisfied: wrapt>=1.11.0 in c:\programdata\anaconda3\lib
\site-packages (from tensorflow) (1.12.1)
Requirement already satisfied: protobuf<3.20,>=3.9.2 in c:\programdata\anacon
da3\lib\site-packages (from tensorflow) (3.19.1)
Requirement already satisfied: opt-einsum>=2.3.2 in c:\programdata\anaconda3
\lib\site-packages (from tensorflow) (3.3.0)
Requirement already satisfied: typing-extensions>=3.6.6 in c:\programdata\ana
conda3\lib\site-packages (from tensorflow) (4.1.1)
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in c:\programdata\anaconda
3\lib\site-packages (from tensorflow) (0.4.0)
Requirement already satisfied: termcolor>=1.1.0 in c:\programdata\anaconda3\l
ib\site-packages (from tensorflow) (1.1.0)
Requirement already satisfied: libclang>=13.0.0 in c:\programdata\anaconda3\l
ib\site-packages (from tensorflow) (14.0.1)
Requirement already satisfied: astunparse>=1.6.0 in c:\programdata\anaconda3
\lib\site-packages (from tensorflow) (1.6.3)
Requirement already satisfied: keras-preprocessing>=1.1.1 in c:\programdata\a
naconda3\lib\site-packages (from tensorflow) (1.1.2)
Requirement already satisfied: absl-py>=1.0.0 in c:\programdata\anaconda3\lib
\site-packages (from tensorflow) (1.1.0)
Requirement already satisfied: tensorboard<2.10,>=2.9 in c:\programdata\anaco
nda3\lib\site-packages (from tensorflow) (2.9.1)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in c:\programdata\anaconda
3\lib\site-packages (from tensorflow) (1.42.0)
Requirement already satisfied: flatbuffers<2,>=1.12 in c:\programdata\anacond
a3\lib\site-packages (from tensorflow) (1.12)
Requirement already satisfied: setuptools in c:\programdata\anaconda3\lib\sit
e-packages (from tensorflow) (61.2.0)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in c:\pro
gramdata\anaconda3\lib\site-packages (from tensorflow) (0.26.0)
Requirement already satisfied: tensorflow-estimator<2.10.0,>=2.9.0rc0 in c:\p
rogramdata\anaconda3\lib\site-packages (from tensorflow) (2.9.0)
Requirement already satisfied: keras<2.10.0,>=2.9.0rc0 in c:\programdata\anac
onda3\lib\site-packages (from tensorflow) (2.9.0)
Requirement already satisfied: numpy>=1.20 in c:\programdata\anaconda3\lib\si
te-packages (from tensorflow) (1.21.5)
Requirement already satisfied: h5py>=2.9.0 in c:\programdata\anaconda3\lib\si
te-packages (from tensorflow) (3.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in c:\programdata\anaconda
3\lib\site-packages (from tensorflow) (0.2.0)
Requirement already satisfied: six>=1.12.0 in c:\programdata\anaconda3\lib\si
te-packages (from tensorflow) (1.16.0)
Requirement already satisfied: packaging in c:\programdata\anaconda3\lib\site
-packages (from tensorflow) (21.3)
Requirement already satisfied: wheel<1.0,>=0.23.0 in c:\programdata\anaconda3
\lib\site-packages (from astunparse>=1.6.0->tensorflow) (0.37.1)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in c:\programdat
a\anaconda3\lib\site-packages (from tensorboard<2.10,>=2.9->tensorflow) (1.8.
1)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in c:\program
data\anaconda3\lib\site-packages (from tensorboard<2.10,>=2.9->tensorflow)
(0.4.6)
Requirement already satisfied: markdown>=2.6.8 in c:\programdata\anaconda3\li
```

```
b\site-packages (from tensorboard<2.10,>=2.9->tensorflow) (3.3.4)
Requirement already satisfied: requests<3,>=2.21.0 in c:\programdata\anaconda
3\lib\site-packages (from tensorboard<2.10,>=2.9->tensorflow) (2.27.1)
Requirement already satisfied: werkzeug>=1.0.1 in c:\programdata\anaconda3\li
b\site-packages (from tensorboard<2.10,>=2.9->tensorflow) (2.0.3)
Requirement already satisfied: google-auth<3,>=1.6.3 in c:\programdata\anacon
da3\lib\site-packages (from tensorboard<2.10,>=2.9->tensorflow) (1.33.0)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in c:\pr
ogramdata\anaconda3\lib\site-packages (from tensorboard<2.10,>=2.9->tensorflo
w) (0.6.1)
Requirement already satisfied: rsa<5,>=3.1.4 in c:\programdata\anaconda3\lib
\site-packages (from google-auth<3,>=1.6.3->tensorboard<2.10,>=2.9->tensorflo
w) (4.7.2)
Requirement already satisfied: cachetools<5.0,>=2.0.0 in c:\programdata\anaco
nda3\lib\site-packages (from google-auth<3,>=1.6.3->tensorboard<2.10,>=2.9->t
ensorflow) (4.2.2)
Requirement already satisfied: pyasn1-modules>=0.2.1 in c:\programdata\anacon
da3\lib\site-packages (from google-auth<3,>=1.6.3->tensorboard<2.10,>=2.9->te
nsorflow) (0.2.8)
Requirement already satisfied: requests-oauthlib>=0.7.0 in c:\programdata\ana
conda3\lib\site-packages (from google-auth-oauthlib<0.5,>=0.4.1->tensorboard<
2.10,>=2.9->tensorflow) (1.3.1)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in c:\programdata\anacond
a3\lib\site-packages (from pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->tens
orboard<2.10,>=2.9->tensorflow) (0.4.8)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\programdata\anacon
da3\lib\site-packages (from requests<3,>=2.21.0->tensorboard<2.10,>=2.9->tens
orflow) (1.26.9)
Requirement already satisfied: charset-normalizer~=2.0.0 in c:\programdata\an
aconda3\lib\site-packages (from requests<3,>=2.21.0->tensorboard<2.10,>=2.9->
tensorflow) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\programdata\anaconda3\lib\s
ite-packages (from requests<3,>=2.21.0->tensorboard<2.10,>=2.9->tensorflow)
(3.3)
Requirement already satisfied: certifi>=2017.4.17 in c:\programdata\anaconda3
\lib\site-packages (from requests<3,>=2.21.0->tensorboard<2.10,>=2.9->tensorf
low) (2021.10.8)
Requirement already satisfied: oauthlib>=3.0.0 in c:\programdata\anaconda3\li
b\site-packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<0.5,>=0.
4.1->tensorboard<2.10,>=2.9->tensorflow) (3.2.0)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\programdata\ana
conda3\lib\site-packages (from packaging->tensorflow) (3.0.4)
Note: you may need to restart the kernel to use updated packages.
```

In [3]:
```
pip install sklearn
```

```
Requirement already satisfied: sklearn in c:\programdata\anaconda3\lib\site-pac
kages (0.0)
Requirement already satisfied: scikit-learn in c:\programdata\anaconda3\lib\sit
e-packages (from sklearn) (1.0.2)
Requirement already satisfied: scipy>=1.1.0 in c:\programdata\anaconda3\lib\sit
e-packages (from scikit-learn->sklearn) (1.7.3)
Requirement already satisfied: joblib>=0.11 in c:\programdata\anaconda3\lib\sit
e-packages (from scikit-learn->sklearn) (1.1.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\programdata\anaconda3
\lib\site-packages (from scikit-learn->sklearn) (2.2.0)
Requirement already satisfied: numpy>=1.14.6 in c:\programdata\anaconda3\lib\si
te-packages (from scikit-learn->sklearn) (1.21.5)
Note: you may need to restart the kernel to use updated packages.
```

In [4]:
```
pip install matplotlib
```

```
Requirement already satisfied: matplotlib in c:\programdata\anaconda3\lib\site-
packages (3.5.1)
Requirement already satisfied: python-dateutil>=2.7 in c:\programdata\anaconda3
\lib\site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: cycler>=0.10 in c:\programdata\anaconda3\lib\sit
e-packages (from matplotlib) (0.11.0)
Requirement already satisfied: pyparsing>=2.2.1 in c:\programdata\anaconda3\lib
\site-packages (from matplotlib) (3.0.4)
Requirement already satisfied: packaging>=20.0 in c:\programdata\anaconda3\lib
\site-packages (from matplotlib) (21.3)
Requirement already satisfied: pillow>=6.2.0 in c:\programdata\anaconda3\lib\si
te-packages (from matplotlib) (9.0.1)
Requirement already satisfied: numpy>=1.17 in c:\programdata\anaconda3\lib\site
-packages (from matplotlib) (1.21.5)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\programdata\anaconda3\li
b\site-packages (from matplotlib) (1.3.2)
Requirement already satisfied: fonttools>=4.22.0 in c:\programdata\anaconda3\li
b\site-packages (from matplotlib) (4.25.0)
Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\lib\site-pa
ckages (from python-dateutil>=2.7->matplotlib) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [5]:
```
pip install keras
```

```
Requirement already satisfied: keras in c:\programdata\anaconda3\lib\site-packa
ges (2.9.0)
Note: you may need to restart the kernel to use updated packages.
```

In [6]:
```
pip install seaborn
```

```
Requirement already satisfied: seaborn in c:\programdata\anaconda3\lib\site-pac
kages (0.11.2)
Requirement already satisfied: scipy>=1.0 in c:\programdata\anaconda3\lib\site-
packages (from seaborn) (1.7.3)
Requirement already satisfied: numpy>=1.15 in c:\programdata\anaconda3\lib\site
-packages (from seaborn) (1.21.5)
Requirement already satisfied: pandas>=0.23 in c:\programdata\anaconda3\lib\sit
e-packages (from seaborn) (1.4.2)
Requirement already satisfied: matplotlib>=2.2 in c:\programdata\anaconda3\lib
\site-packages (from seaborn) (3.5.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\programdata\anaconda3\li
b\site-packages (from matplotlib>=2.2->seaborn) (4.25.0)
Requirement already satisfied: packaging>=20.0 in c:\programdata\anaconda3\lib
\site-packages (from matplotlib>=2.2->seaborn) (21.3)
Requirement already satisfied: python-dateutil>=2.7 in c:\programdata\anaconda3
\lib\site-packages (from matplotlib>=2.2->seaborn) (2.8.2)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\programdata\anaconda3\li
b\site-packages (from matplotlib>=2.2->seaborn) (1.3.2)
Requirement already satisfied: cycler>=0.10 in c:\programdata\anaconda3\lib\sit
e-packages (from matplotlib>=2.2->seaborn) (0.11.0)
Requirement already satisfied: pyparsing>=2.2.1 in c:\programdata\anaconda3\lib
\site-packages (from matplotlib>=2.2->seaborn) (3.0.4)
Requirement already satisfied: pillow>=6.2.0 in c:\programdata\anaconda3\lib\si
te-packages (from matplotlib>=2.2->seaborn) (9.0.1)
Requirement already satisfied: pytz>=2020.1 in c:\programdata\anaconda3\lib\sit
e-packages (from pandas>=0.23->seaborn) (2021.3)
Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\lib\site-pa
ckages (from python-dateutil>=2.7->matplotlib>=2.2->seaborn) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [7]:
```
pip install pydotplus
```

```
Requirement already satisfied: pydotplus in c:\programdata\anaconda3\lib\site-p
ackages (2.0.2)
Requirement already satisfied: pyparsing>=2.0.1 in c:\programdata\anaconda3\lib
\site-packages (from pydotplus) (3.0.4)
Note: you may need to restart the kernel to use updated packages.
```

In [8]:

```
import matplotlib.pyplot as plt
from PIL import Image
import seaborn as sns
import numpy as np
import pandas as pd
import os
from tensorflow.keras.utils import to_categorical
from glob import glob


from sklearn.model_selection import train_test_split
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout
import tensorflow as tf
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_repo
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ReduceLROnPlateau
from tensorflow.keras import layers
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten

from sklearn.metrics import confusion_matrix
import itertools
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings("ignore")
```

In [9]:

```
df = pd.read_csv('skinCancer/HAM10000_metadata.csv')
df.head(10)
```

Out[9]:

	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
5	HAM_0001466	ISIC_0027850	bkl	histo	75.0	male	ear
6	HAM_0002761	ISIC_0029176	bkl	histo	60.0	male	face
7	HAM_0002761	ISIC_0029068	bkl	histo	60.0	male	face
8	HAM_0005132	ISIC_0025837	bkl	histo	70.0	female	back
9	HAM_0005132	ISIC_0025209	bkl	histo	70.0	female	back

In [10]:
```
df.isnull().sum()
```

Out[10]:
```
lesion_id        0
image_id         0
dx               0
dx_type          0
age             57
sex              0
localization     0
dtype: int64
```

In [11]:
```
df.count()
```

Out[11]:
```
lesion_id       10015
image_id        10015
dx              10015
dx_type         10015
age              9958
sex             10015
localization    10015
dtype: int64
```

In [12]:
```
df['age'].fillna(int(df['age'].mean()),inplace=True)
df.isnull().sum()
```

Out[12]:
```
lesion_id       0
image_id        0
dx              0
dx_type         0
age             0
sex             0
localization    0
dtype: int64
```

In [13]:
```
lesion_type_dict = {
    'nv': 'Melanocytic nevi',
    'mel': 'Melanoma',
    'bkl': 'Benign keratosis-like lesions ',
    'bcc': 'Basal cell carcinoma',
    'akiec': 'Actinic keratoses',
    'vasc': 'Vascular lesions',
    'df': 'Dermatofibroma'
}
base_skin_dir = 'skinCancer'

imageid_path_dict = {os.path.splitext(os.path.basename(x))[0]: x
                     for x in glob(os.path.join(base_skin_dir, '*', '*.jpg'))}
```

In [14]:
```
df['path'] = df['image_id'].map(imageid_path_dict.get)
df['cell_type'] = df['dx'].map(lesion_type_dict.get)
df['cell_type_idx'] = pd.Categorical(df['cell_type']).codes
df.head()
```

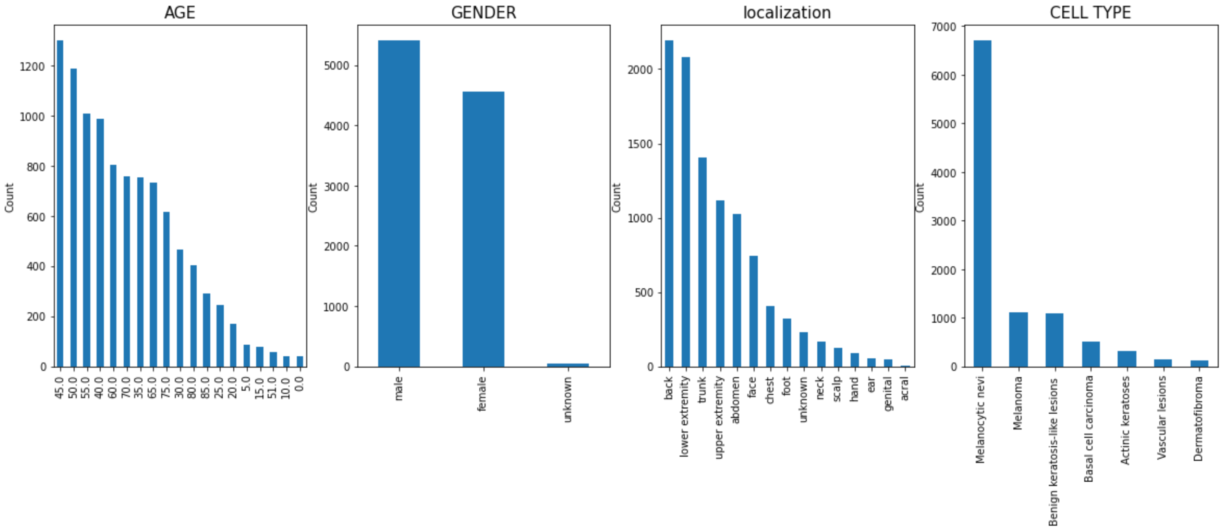
Out[14]:

	lesion_id	image_id	dx	dx_type	age	sex	localization	
0	HAM_0000118	ISIC_0027419	bkl	histo	80.0	male	scalp	skinCancer\HAM10000_images_
1	HAM_0000118	ISIC_0025030	bkl	histo	80.0	male	scalp	skinCancer\HAM10000_images_
2	HAM_0002730	ISIC_0026769	bkl	histo	80.0	male	scalp	skinCancer\HAM10000_images_
3	HAM_0002730	ISIC_0025661	bkl	histo	80.0	male	scalp	skinCancer\HAM10000_images_
4	HAM_0001466	ISIC_0031633	bkl	histo	75.0	male	ear	skinCancer\HAM10000_images_

In [15]:
```
df['image'] = df['path'].map(lambda x: np.asarray(Image.open(x).resize((125,100))
```

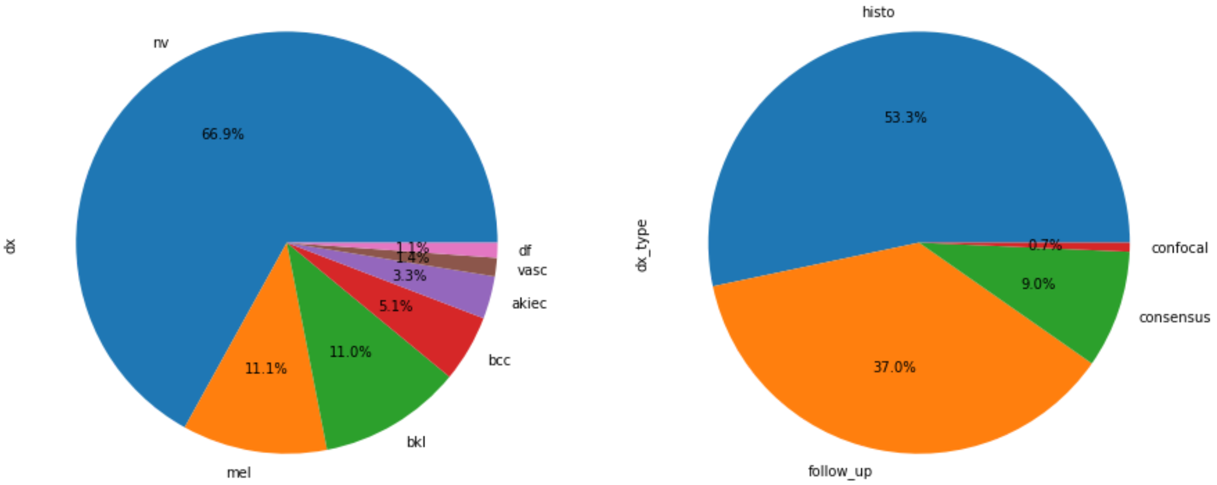
In [16]:
```
plt.figure(figsize=(20,10))
plt.subplots_adjust(left=0.125, bottom=1, right=0.9, top=2, hspace=0.2)
plt.subplot(2,4,1)
plt.title("AGE",fontsize=15)
plt.ylabel("Count")
df['age'].value_counts().plot.bar()

plt.subplot(2,4,2)
plt.title("GENDER",fontsize=15)
plt.ylabel("Count")
df['sex'].value_counts().plot.bar()

plt.subplot(2,4,3)
plt.title("localization",fontsize=15)
plt.ylabel("Count")
plt.xticks(rotation=45)
df['localization'].value_counts().plot.bar()

plt.subplot(2,4,4)
plt.title("CELL TYPE",fontsize=15)
plt.ylabel("Count")
df['cell_type'].value_counts().plot.bar()
```

Out[16]:
```
<AxesSubplot:title={'center':'CELL TYPE'}, ylabel='Count'>
```



In [17]:

```
plt.figure(figsize=(15,10))
plt.subplot(1,2,1)
df['dx'].value_counts().plot.pie(autopct="%1.1f%%")
plt.subplot(1,2,2)
df['dx_type'].value_counts().plot.pie(autopct="%1.1f%%")
plt.show()
```



In [18]:

```
features=df.drop(columns=['cell_type_idx','image_id'],axis=1)
target=df['cell_type_idx']
features.head()
```

Out[18]:

	lesion_id	dx	dx_type	age	sex	localization	
0	HAM_0000118	bkl	histo	80.0	male	scalp	skinCancer\HAM10000_images_part_1\ISIC_002
1	HAM_0000118	bkl	histo	80.0	male	scalp	skinCancer\HAM10000_images_part_1\ISIC_002
2	HAM_0002730	bkl	histo	80.0	male	scalp	skinCancer\HAM10000_images_part_1\ISIC_002
3	HAM_0002730	bkl	histo	80.0	male	scalp	skinCancer\HAM10000_images_part_1\ISIC_002
4	HAM_0001466	bkl	histo	75.0	male	ear	skinCancer\HAM10000_images_part_2\ISIC_003

In [19]:
```
x_train_o, x_test_o, y_train_o, y_test_o = train_test_split(features, target, tes
tf.unique(x_train_o.cell_type.values)
```

Out[19]:
```
Unique(y=<tf.Tensor: shape=(7,), dtype=string, numpy=
array([b'Melanocytic nevi', b'Basal cell carcinoma', b'Melanoma',
       b'Vascular lesions', b'Benign keratosis-like lesions ',
       b'Actinic keratoses', b'Dermatofibroma'], dtype=object)>, idx=<tf.Tenso
r: shape=(7511,), dtype=int32, numpy=array([0, 1, 0, ..., 1, 0, 0])>)
```

In [20]:
```
x_train = np.asarray(x_train_o['image'].tolist())
x_test = np.asarray(x_test_o['image'].tolist())

x_train_mean = np.mean(x_train)
x_train_std = np.std(x_train)


x_test_mean = np.mean(x_test)
x_test_std = np.std(x_test)

x_train = (x_train - x_train_mean)/x_train_std
x_test = (x_test - x_test_mean)/x_test_std
```

In [21]:
```
# Perform one-hot encoding on the labels
y_train = to_categorical(y_train_o, num_classes = 7)
y_test = to_categorical(y_test_o, num_classes = 7)
y_test
```

Out[21]:
```
array([[0., 0., 0., ..., 0., 1., 0.],
       [0., 0., 0., ..., 1., 0., 0.],
       [1., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 1., 0., 0.],
       [0., 0., 0., ..., 1., 0., 0.],
       [0., 0., 0., ..., 1., 0., 0.]], dtype=float32)
```

In [22]:
```
y_test[1]
```

Out[22]:
```
array([0., 0., 0., 0., 1., 0., 0.], dtype=float32)
```

In [23]:
```
y_train
```

Out[23]:
```
array([[0., 0., 0., ..., 1., 0., 0.],
       [0., 1., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 1., 0., 0.],
       ...,
       [0., 1., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 1., 0., 0.],
       [0., 0., 0., ..., 1., 0., 0.]], dtype=float32)
```

In [24]:
```
y_test_o.value_counts()
```

Out[24]:
```
4    1693
2     277
5     274
1     123
0      68
6      39
3      30
Name: cell_type_idx, dtype: int64
```

In [25]:
```
# x_train, x_validate, y_train, y_validate = train_test_split(x_train, y_train, t


# Reshape image in 3 dimensions (height = 100, width = 125 , canal = 3)
x_train = x_train.reshape(x_train_o.shape[0], *(100, 125, 3))
x_test = x_test.reshape(x_test_o.shape[0], *(100, 125, 3))
```

In [26]:
```
x_train = x_train.reshape(x_train.shape[0],125*100*3)
x_test = x_test.reshape(x_test.shape[0],125*100*3)
print(x_train.shape)
print(x_test.shape)
```

```
(7511, 37500)
(2504, 37500)
```

In [27]:
```
print(x_train)
```

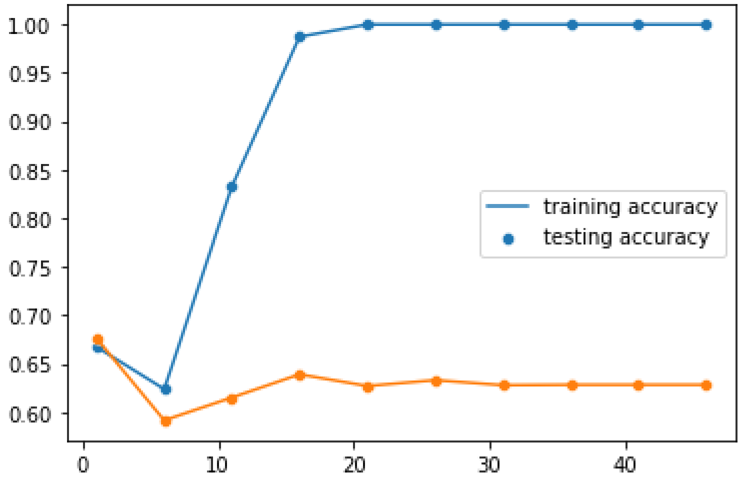
```
[[ 1.55231607e+00 -3.87567404e-01 -1.72024796e-01 ...  9.27242505e-01
  -5.60001490e-01 -6.03110012e-01]
 [ 3.23723203e-01 -4.09121664e-01 -1.72024796e-01 ...  3.66831724e-01
  -3.44458882e-01 -1.07362013e-01]
 [ 1.22900216e+00 -7.97098359e-01 -2.15133317e-01 ...  1.09967659e+00
  -3.87567404e-01 -2.15133317e-01]
 ...
 [-2.56454774e+00 -2.90941592e+00 -2.43522218e+00 ... -2.78009035e+00
  -2.99563296e+00 -2.65076479e+00]
 [ 4.53048767e-01 -1.07362013e-01 -8.58077525e-02 ...  3.88385985e-01
  -4.26992309e-02  4.09290704e-04]
 [ 1.63853311e+00 -3.01350360e-01 -2.15133317e-01 ...  1.09967659e+00
  -4.52230186e-01 -6.46218533e-01]]
```

Decision Tree

In [28]:
```python
depth = range(1,51,5)
testing_accuracy = []
training_accuracy = []
score = 0

for i in depth:
    tree = DecisionTreeClassifier(max_depth = i, criterion = 'entropy')
    tree.fit(x_train, y_train)

    y_predict_train = tree.predict(x_train)
    training_accuracy.append(accuracy_score(y_train, y_predict_train))

    y_predict_test = tree.predict(x_test)
    acc_score = accuracy_score(y_test,y_predict_test)
    testing_accuracy.append(acc_score)

    print(i)

    if score < acc_score:
        score = acc_score
        best_depth = i

sns.lineplot(depth, training_accuracy)
sns.scatterplot(depth, training_accuracy)
sns.lineplot(depth, testing_accuracy)
sns.scatterplot(depth, testing_accuracy)
plt.legend(['training accuracy', 'testing accuracy'])
```

```
1
6
11
16
21
26
31
36
41
46
```

Out[28]:
```
<matplotlib.legend.Legend at 0x23052cff760>
```



In [29]:
```
print('This is the best depth for Decision Tree Classifier: ', best_depth, '\nAcc
result = confusion_matrix(y_test.argmax(axis=1), y_predict_test.argmax(axis=1),)
print("Confusion Matrix:","\n", result)
report = classification_report(y_test.argmax(axis=1), y_predict_test.argmax(axis=
print("Classification Report:" , "\n", report)
```

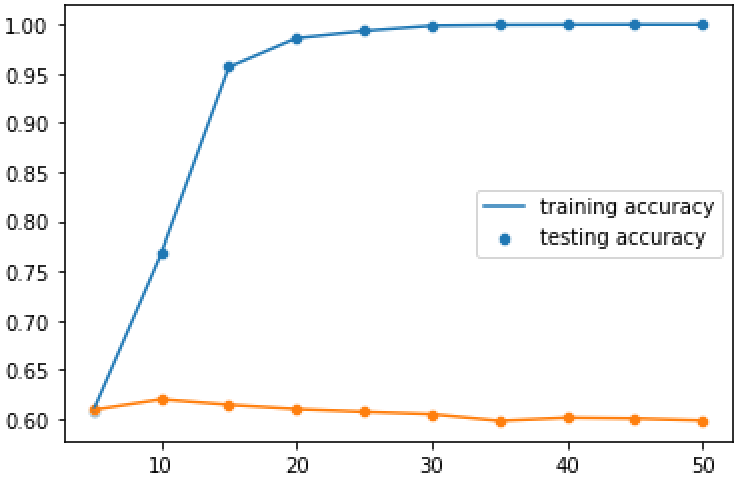
```
This is the best depth for Decision Tree Classifier:  1
Accuracy score is:  0.6761182108626198
Confusion Matrix:
 [[  12   15   14    3   18    5    1]
 [  15   30   25    6   34    9    4]
 [  12   21   94    6  102   41    1]
 [   3    5    6    1   11    3    1]
 [  18   37  111   10 1357  149   11]
 [   8   14   49    0  118   77    8]
 [   3    4    7    0   20    3    2]]
Classification Report:
               precision    recall  f1-score   support

           0       0.17      0.18      0.17        68
           1       0.24      0.24      0.24       123
           2       0.31      0.34      0.32       277
           3       0.04      0.03      0.04        30
           4       0.82      0.80      0.81      1693
           5       0.27      0.28      0.27       274
           6       0.07      0.05      0.06        39

    accuracy                           0.63      2504
   macro avg       0.27      0.28      0.27      2504
weighted avg       0.63      0.63      0.63      2504
```

Random Forest

In [30]:
```
#random Forest
depth = range(5,51,5)
testing_accuracy = []
training_accuracy = []
score = 0

for i in depth:
    tree = RandomForestClassifier(max_depth = i, criterion = 'gini', random_state
    tree.fit(x_train, y_train)

    y_predict_train = tree.predict(x_train)
    training_accuracy.append(accuracy_score(y_train, y_predict_train))

    y_predict_test = tree.predict(x_test)
    acc_score = accuracy_score(y_test,y_predict_test)
    testing_accuracy.append(acc_score)

    print(i)

    if score < acc_score:
        score = acc_score
        best_depth = i

sns.lineplot(depth, training_accuracy)
sns.scatterplot(depth, training_accuracy)
sns.lineplot(depth, testing_accuracy)
sns.scatterplot(depth, testing_accuracy)
plt.legend(['training accuracy', 'testing accuracy'])
```

```
5
10
15
20
25
30
35
40
45
50
```

Out[30]:
```
<matplotlib.legend.Legend at 0x23053202880>
```



In [31]:

```
print('This is the best depth for Decision Tree Classifier: ', best_depth, '\nAcc
result = confusion_matrix(y_test.argmax(axis=1), y_predict_test.argmax(axis=1),)
print("Confusion Matrix:","\n", result)
report = classification_report(y_test.argmax(axis=1), y_predict_test.argmax(axis=
print("Classification Report:" , "\n", report)
```

```
This is the best depth for Decision Tree Classifier:  10
Accuracy score is:  0.6202076677316294
Confusion Matrix:
 [[  63    1    0    0    4    0    0]
 [  99    7    2    0   15    0    0]
 [ 164    0   25    0   88    0    0]
 [  23    0    0    0    7    0    0]
 [ 221    0   10    0 1459    3    0]
 [ 149    0    3    0  114    8    0]
 [  21    0    0    0   18    0    0]]
Classification Report:
               precision    recall  f1-score   support

           0       0.09      0.93      0.16        68
           1       0.88      0.06      0.11       123
           2       0.62      0.09      0.16       277
           3       0.00      0.00      0.00        30
           4       0.86      0.86      0.86      1693
           5       0.73      0.03      0.06       274
           6       0.00      0.00      0.00        39

    accuracy                           0.62      2504
   macro avg       0.45      0.28      0.19      2504
weighted avg       0.77      0.62      0.61      2504
```

KNN

In [32]:
```
k = range(1,500,2)
testing_accuracy = []
training_accuracy = []
score = 0

for i in k:
    knn = KNeighborsClassifier(n_neighbors = i)
    knn.fit(x_train, y_train)

    y_predict_train = knn.predict(x_train)
    training_accuracy.append(accuracy_score(y_train, y_predict_train))

    y_predict_test = knn.predict(x_test)
    acc_score = accuracy_score(y_test,y_predict_test)
    testing_accuracy.append(acc_score)

    print(i)

    if score < acc_score:
        score = acc_score
        best_k = i


sns.lineplot(k, training_accuracy)
sns.scatterplot(k, training_accuracy)
sns.lineplot(k, testing_accuracy)
sns.scatterplot(k, testing_accuracy)
plt.legend(['training accuracy', 'testing accuracy'])
```

```
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
35
37
```

In [33]:
```
'''print('This is the best depth for Decision Tree Classifier: ', best_depth, '\r
result = confusion_matrix(y_test.argmax(axis=1), y_predict_test.argmax(axis=1),)
print("Confusion Matrix:","\n", result)
report = classification_report(y_test.argmax(axis=1), y_predict_test.argmax(axis=
print("Classification Report:" , "\n", report)
```

```
Confusion Matrix:
 [[  45    0    0    0   23    0    0]
 [  70    0    0    0   53    0    0]
 [ 140    0    0    0  137    0    0]
 [  13    0    0    0   17    0    0]
 [ 141    0    0    0 1552    0    0]
 [  80    0    0    0  194    0    0]
 [  11    0    0    0   28    0    0]]
Classification Report:
               precision    recall  f1-score   support

           0       0.09      0.66      0.16        68
           1       0.00      0.00      0.00       123
           2       0.00      0.00      0.00       277
           3       0.00      0.00      0.00        30
           4       0.77      0.92      0.84      1693
           5       0.00      0.00      0.00       274
           6       0.00      0.00      0.00        39

    accuracy                           0.64      2504
   macro avg       0.12      0.23      0.14      2504
weighted avg       0.53      0.64      0.57      2504
```

In [34]:
```
x_train = x_train.reshape(x_train_o.shape[0], *(100, 125, 3))
x_test = x_test.reshape(x_test_o.shape[0], *(100, 125, 3))
x_train.shape
```

Out[34]:
```
(7511, 100, 125, 3)
```

MLP

In [35]:

```
x_train = x_train.reshape(x_train.shape[0],125*100*3)
x_test = x_test.reshape(x_test.shape[0],125*100*3)

# define the keras model
model = Sequential()

model.add(Dense(units= 128, kernel_initializer = 'uniform', activation = 'relu',
model.add(Dense(units= 256, kernel_initializer = 'uniform', activation = 'relu'))
model.add(Dense(units= 512, kernel_initializer = 'uniform', activation = 'relu'))
model.add(Dense(units= 64, kernel_initializer = 'uniform', activation = 'relu'))
model.add(Dense(units = 7, kernel_initializer = 'uniform', activation = 'softmax'
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense (Dense)               (None, 128)               4800128   
                                                                 
 dense_1 (Dense)             (None, 256)               33024     
                                                                 
 dense_2 (Dense)             (None, 512)               131584    
                                                                 
 dense_3 (Dense)             (None, 64)                32832     
                                                                 
 dense_4 (Dense)             (None, 7)                 455       
                                                                 
=================================================================
Total params: 4,998,023
Trainable params: 4,998,023
Non-trainable params: 0
_________________________________________________________________
```

In [36]:

```
# compile the keras model
model.compile(optimizer = "Adam", loss = 'categorical_crossentropy', metrics = ['

# fit the keras model on the dataset
history = model.fit(x_train, y_train, batch_size = 500, epochs = 25)
```

```
Epoch 1/25
16/16 [==============================] - 1s 39ms/step - loss: 1.1285 - accurac
y: 0.6287
Epoch 2/25
16/16 [==============================] - 1s 38ms/step - loss: 0.9556 - accurac
y: 0.6684
Epoch 3/25
16/16 [==============================] - 1s 39ms/step - loss: 0.9155 - accurac
y: 0.6799
Epoch 4/25
16/16 [==============================] - 1s 40ms/step - loss: 0.8785 - accurac
y: 0.6899
Epoch 5/25
16/16 [==============================] - 1s 39ms/step - loss: 0.8875 - accurac
y: 0.6889
Epoch 6/25
16/16 [==============================] - 1s 39ms/step - loss: 0.8520 - accurac
y: 0.7026
Epoch 7/25
16/16 [==============================] - 1s 39ms/step - loss: 0.8204 - accurac
y: 0.7043
Epoch 8/25
16/16 [==============================] - 1s 39ms/step - loss: 0.8002 - accurac
y: 0.7165
Epoch 9/25
16/16 [==============================] - 1s 39ms/step - loss: 0.8057 - accurac
y: 0.7122
Epoch 10/25
16/16 [==============================] - 1s 40ms/step - loss: 0.7512 - accurac
y: 0.7301
Epoch 11/25
16/16 [==============================] - 1s 39ms/step - loss: 0.7543 - accurac
y: 0.7292
Epoch 12/25
16/16 [==============================] - 1s 39ms/step - loss: 0.7248 - accurac
y: 0.7343
Epoch 13/25
16/16 [==============================] - 1s 40ms/step - loss: 0.7436 - accurac
y: 0.7321
Epoch 14/25
16/16 [==============================] - 1s 40ms/step - loss: 0.7293 - accurac
y: 0.7430
Epoch 15/25
16/16 [==============================] - 1s 39ms/step - loss: 0.7396 - accurac
y: 0.7339
Epoch 16/25
16/16 [==============================] - 1s 40ms/step - loss: 0.6752 - accurac
y: 0.7488
Epoch 17/25
16/16 [==============================] - 1s 39ms/step - loss: 0.7258 - accurac
y: 0.7402
```

```
Epoch 18/25
16/16 [==============================] - 1s 40ms/step - loss: 0.7219 - accurac
y: 0.7376
Epoch 19/25
16/16 [==============================] - 1s 39ms/step - loss: 0.6495 - accurac
y: 0.7556
Epoch 20/25
16/16 [==============================] - 1s 39ms/step - loss: 0.6136 - accurac
y: 0.7729
Epoch 21/25
16/16 [==============================] - 1s 39ms/step - loss: 0.6019 - accurac
y: 0.7755
Epoch 22/25
16/16 [==============================] - 1s 39ms/step - loss: 0.5660 - accurac
y: 0.7880
Epoch 23/25
16/16 [==============================] - 1s 39ms/step - loss: 0.5493 - accurac
y: 0.7952
Epoch 24/25
16/16 [==============================] - 1s 39ms/step - loss: 0.6436 - accurac
y: 0.7637
Epoch 25/25
16/16 [==============================] - 1s 38ms/step - loss: 0.6152 - accurac
y: 0.7705
```

In [37]:

```
accuracy = model.evaluate(x_test, y_test, verbose=1)[1]
print("Test: accuracy = ",accuracy*100,"%")
```

```
79/79 [==============================] - 0s 3ms/step - loss: 0.8449 - accuracy:
0.7077
Test: accuracy =  70.76677083969116 %
```

CNN

In [38]:

```
x_train = x_train.reshape(x_train.shape[0], 125,100,3)
x_test = x_test.reshape(x_test.shape[0], 125, 100, 3)
print(x_train.shape)
print(x_test.shape)
```

```
(7511, 125, 100, 3)
(2504, 125, 100, 3)
```

In [39]:

```
input_shape = (125, 100, 3)
num_classes = 7

model = Sequential()
model.add(Conv2D(64, kernel_size=(3, 3),activation='relu',padding = 'Same',input_
model.add(Conv2D(64,kernel_size=(3, 3), activation='relu',padding = 'Same'))
model.add(MaxPooling2D(pool_size = (2, 2)))
model.add(Dropout(0.16))

model.add(Conv2D(128, (3, 3), activation='relu',padding = 'same'))
model.add(Conv2D(128, (3, 3), activation='relu',padding = 'Same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.20))

model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(num_classes, activation='softmax'))
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 125, 100, 64)      1792      
                                                                 
 conv2d_1 (Conv2D)           (None, 125, 100, 64)      36928     
                                                                 
 max_pooling2d (MaxPooling2D  (None, 62, 50, 64)       0         
 )                                                               
                                                                 
 dropout (Dropout)           (None, 62, 50, 64)        0         
                                                                 
 conv2d_2 (Conv2D)           (None, 62, 50, 128)       73856     
                                                                 
 conv2d_3 (Conv2D)           (None, 62, 50, 128)       147584    
                                                                 
 max_pooling2d_1 (MaxPooling  (None, 31, 25, 128)      0         
 2D)                                                             
                                                                 
 dropout_1 (Dropout)         (None, 31, 25, 128)       0         
                                                                 
 flatten (Flatten)           (None, 99200)             0         
                                                                 
 dense_5 (Dense)             (None, 512)               50790912  
                                                                 
 dense_6 (Dense)             (None, 256)               131328    
                                                                 
 dropout_2 (Dropout)         (None, 256)               0         
                                                                 
 dense_7 (Dense)             (None, 7)                 1799      
                                                                 
=================================================================
Total params: 51,184,199
Trainable params: 51,184,199
```

```
Non-trainable params: 0
_________________________________________________________________
```

In [40]:

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

learning_rate_reduction = ReduceLROnPlateau(monitor='val_accuracy',
                                            patience=4,
                                            verbose=1,
                                            factor=0.5,
                                            min_lr=0.00001)
```

In [43]:
```
#x_train, x_validate, y_train, y_validate = train_test_split(x_train, y_train, te


# Reshape image in 3 dimensions (height = 100, width = 125 , canal = 3)
x_train = x_train.reshape(x_train.shape[0], *(125, 100, 3))
x_test = x_test.reshape(x_test.shape[0], *(125, 100, 3))
#x_validate = x_validate.reshape(x_validate.shape[0], *(100, 125, 3))

tf.config.run_functions_eagerly(True)


model.fit(x_train, y_train, batch_size = 500, epochs = 25)
```

```
Epoch 1/25
16/16 [==============================] - 265s 16s/step - loss: 0.8516 - accur
acy: 0.6847
Epoch 2/25
16/16 [==============================] - 263s 16s/step - loss: 0.8179 - accur
acy: 0.6986
Epoch 3/25
16/16 [==============================] - 263s 16s/step - loss: 0.7996 - accur
acy: 0.7038
Epoch 4/25
16/16 [==============================] - 264s 16s/step - loss: 0.7991 - accur
acy: 0.7067
Epoch 5/25
16/16 [==============================] - 263s 16s/step - loss: 0.7828 - accur
acy: 0.7084
Epoch 6/25
16/16 [==============================] - 263s 16s/step - loss: 0.7805 - accur
acy: 0.7148
Epoch 7/25
16/16 [==============================] - 263s 16s/step - loss: 0.7621 - accur
acy: 0.7144
Epoch 8/25
16/16 [==============================] - 263s 16s/step - loss: 0.7371 - accur
acy: 0.7244
Epoch 9/25
16/16 [==============================] - 264s 16s/step - loss: 0.7538 - accur
acy: 0.7225
Epoch 10/25
16/16 [==============================] - 264s 16s/step - loss: 0.7293 - accur
acy: 0.7277
Epoch 11/25
16/16 [==============================] - 264s 16s/step - loss: 0.7285 - accur
acy: 0.7237
Epoch 12/25
16/16 [==============================] - 263s 16s/step - loss: 0.7090 - accur
acy: 0.7353
Epoch 13/25
16/16 [==============================] - 264s 16s/step - loss: 0.7153 - accur
acy: 0.7376
Epoch 14/25
16/16 [==============================] - 263s 16s/step - loss: 0.6507 - accur
acy: 0.7564
Epoch 15/25
16/16 [==============================] - 263s 16s/step - loss: 0.7084 - accur
```

```
acy: 0.7405
Epoch 16/25
16/16 [==============================] - 263s 16s/step - loss: 0.6236 - accur
acy: 0.7653
Epoch 17/25
16/16 [==============================] - 265s 16s/step - loss: 0.5971 - accur
acy: 0.7769
Epoch 18/25
16/16 [==============================] - 265s 16s/step - loss: 0.6151 - accur
acy: 0.7670
Epoch 19/25
16/16 [==============================] - 265s 16s/step - loss: 0.5940 - accur
acy: 0.7710
Epoch 20/25
16/16 [==============================] - 264s 16s/step - loss: 0.5401 - accur
acy: 0.8003
Epoch 21/25
16/16 [==============================] - 265s 16s/step - loss: 0.5008 - accur
acy: 0.8095
Epoch 22/25
16/16 [==============================] - 264s 16s/step - loss: 0.4749 - accur
acy: 0.8231
Epoch 23/25
16/16 [==============================] - 264s 16s/step - loss: 0.4405 - accur
acy: 0.8349
Epoch 24/25
16/16 [==============================] - 264s 16s/step - loss: 0.3756 - accur
acy: 0.8601
Epoch 25/25
16/16 [==============================] - 265s 16s/step - loss: 0.3608 - accur
acy: 0.8685
```

Out[43]: `<keras.callbacks.History at 0x23052d94d00>`

In [44]:

```
test_loss, test_acc = model.evaluate(x_test, y_test)
print("Test Accuracy: ",test_acc*100,"%")
```

```
79/79 [==============================] - 23s 285ms/step - loss: 0.9240 - accura
cy: 0.7264
Test Accuracy:  72.64376878738403 %
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: