In [67]:

```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
from imblearn.over_sampling import RandomOverSampler
import numpy as np
from sklearn.model_selection import train_test_split
import os, cv2
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, Flatten, Dense, MaxPool2D
```

In [68]:

```
data = pd.read_csv('/kaggle/input/skin-cancer-mnist-ham10000/hmnist_28_28_RGB.csv')
data.head()
```

Out[68]:

	pixel0000	pixel0001	pixel0002	pixel0003	pixel0004	pixel0005	pixel0006	pixel0007	pixel0008	pixel0009	...	pixel2343	pix
0	192	153	193	195	155	192	197	154	185	202	...	173	
1	25	14	30	68	48	75	123	93	126	158	...	60	
2	192	138	153	200	145	163	201	142	160	206	...	167	
3	38	19	30	95	59	72	143	103	119	171	...	44	
4	158	113	139	194	144	174	215	162	191	225	...	209	

5 rows × 2353 columns

In [69]:

```
data['label'].unique()
```

Out[69]:

```
array([2, 4, 3, 6, 5, 1, 0])
```

In [70]:

```
y = data['label']
x = data.drop(columns = ['label'])
```

In [71]:

```
data.isnull().sum().sum() #no null values present
```

Out[71]:

```
0
```

In [72]:

```
meta_data = pd.read_csv('/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_metadata.csv')
meta_data.head()
```

Out[72]:

	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

	lesion_id	image_id	dx	dx_type	age	sex	localization
3	HAM_0002730	ISIC_0025661	bkl	histo	80.0	male	scalp
4	HAM_0001466	ISIC_0031633	bkl	histo	75.0	male	ear

In [73]:

```
meta_data['dx'].unique()
```

Out[73]:

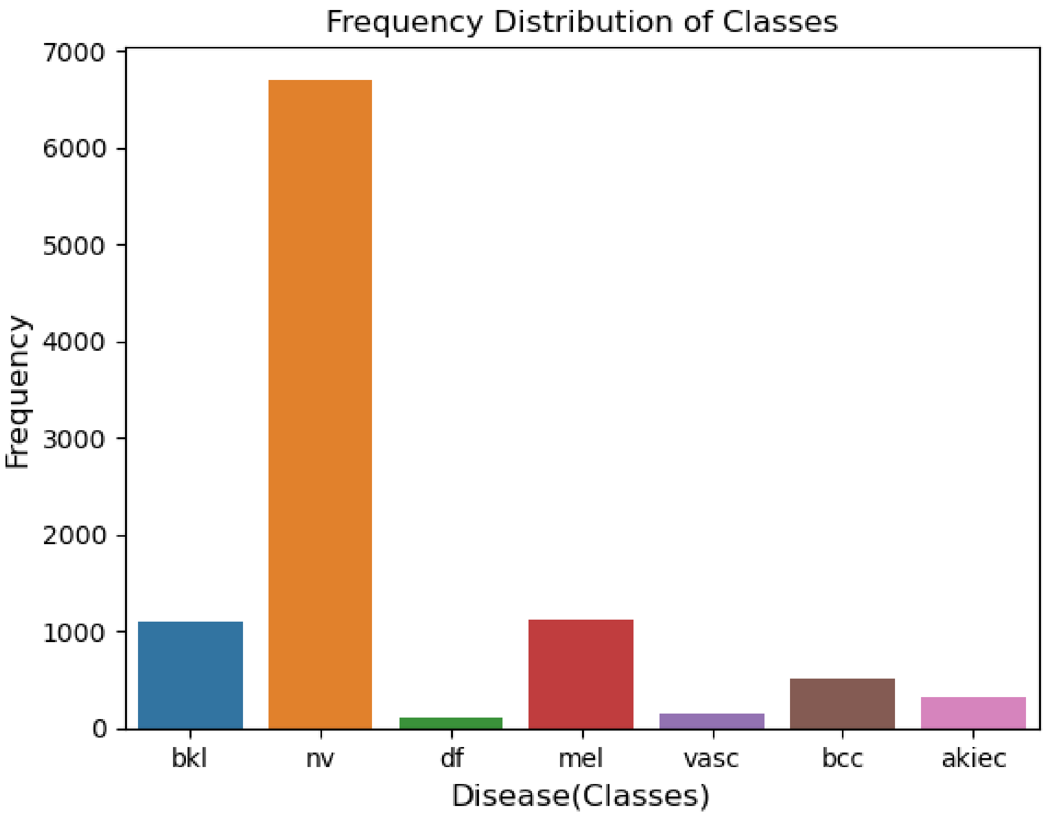
```
array(['bkl', 'nv', 'df', 'mel', 'vasc', 'bcc', 'akiec'], dtype=object)
```

In [74]:

```
sns.countplot(x = 'dx', data = meta_data)
plt.xlabel('Disease(Classes)', size=12)
plt.ylabel('Frequency', size=12)
plt.title('Frequency Distribution of Classes')
```

Out[74]:

```
Text(0.5, 1.0, 'Frequency Distribution of Classes')
```



In [75]:

```
print(x.shape,y.shape)
# To overcome class imbalace
oversample = RandomOverSampler()
x,y  = oversample.fit_resample(x,y)
print(x.shape,y.shape)
```

```
(10015, 2352) (10015,)
(46935, 2352) (46935,)
```

In [ ]:

```
sns.countplot(x = 'dx', data = meta_data)
plt.xlabel('Disease(Classes)', size=12)
plt.ylabel('Frequency', size=12)
plt.title('Frequency Distribution of Classes')
```

In [ ]:

```

```

In [76]:

```
# reshaping the data so that it can be taken by convolution neural network(without distur
bing the no. of samples)
x = np.array(x).reshape(-1,28,28,3)
print('Shape of X :',x.shape)
print('Shape of y :',y.shape)
```

```
Shape of X : (46935, 28, 28, 3)
Shape of y : (46935,)
```

In [77]:

```
# Splitting Data
X_train, X_test, Y_train, Y_test = train_test_split(x,y, test_size=0.2, random_state=1)
print(X_train.shape,Y_train.shape)
print(X_test.shape , Y_test.shape)
```

```
(37548, 28, 28, 3) (37548,)
(9387, 28, 28, 3) (9387,)
```

In [78]:

```
model = Sequential()

model.add(Conv2D(16, kernel_size = (3,3), input_shape = (28, 28, 3), activation = 'relu'
))
model.add(Conv2D(32, kernel_size = (3,3), activation = 'relu'))
model.add(MaxPool2D(pool_size = (2,2)))

model.add(Conv2D(32, kernel_size = (3,3), activation = 'relu'))
model.add(Conv2D(64, kernel_size = (3,3), activation = 'relu'))
model.add(MaxPool2D(pool_size = (2,2)))

model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dense(7, activation='softmax'))
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)           (None, 26, 26, 16)        448

 conv2d_5 (Conv2D)           (None, 24, 24, 32)        4640

 max_pooling2d_2 (MaxPooling  (None, 12, 12, 32)       0
 2D)

 conv2d_6 (Conv2D)           (None, 10, 10, 32)        9248

 conv2d_7 (Conv2D)           (None, 8, 8, 64)          18496

 max_pooling2d_3 (MaxPooling  (None, 4, 4, 64)         0
 2D)

 flatten_1 (Flatten)         (None, 1024)              0

 dense_5 (Dense)             (None, 64)                65600

 dense_6 (Dense)             (None, 7)                 455

=================================================================
Total params: 98,887
Trainable params: 98,887
Non-trainable params: 0
_________________________________________________________________
```

In [79]:

```
model.compile(loss = 'sparse_categorical_crossentropy',
```

```
              optimizer = 'adam',
               metrics = ['accuracy'])
history = model.fit(X_train,
                    Y_train,
                    validation_split=0.2,
                    batch_size = 128,
                    epochs = 50)
```

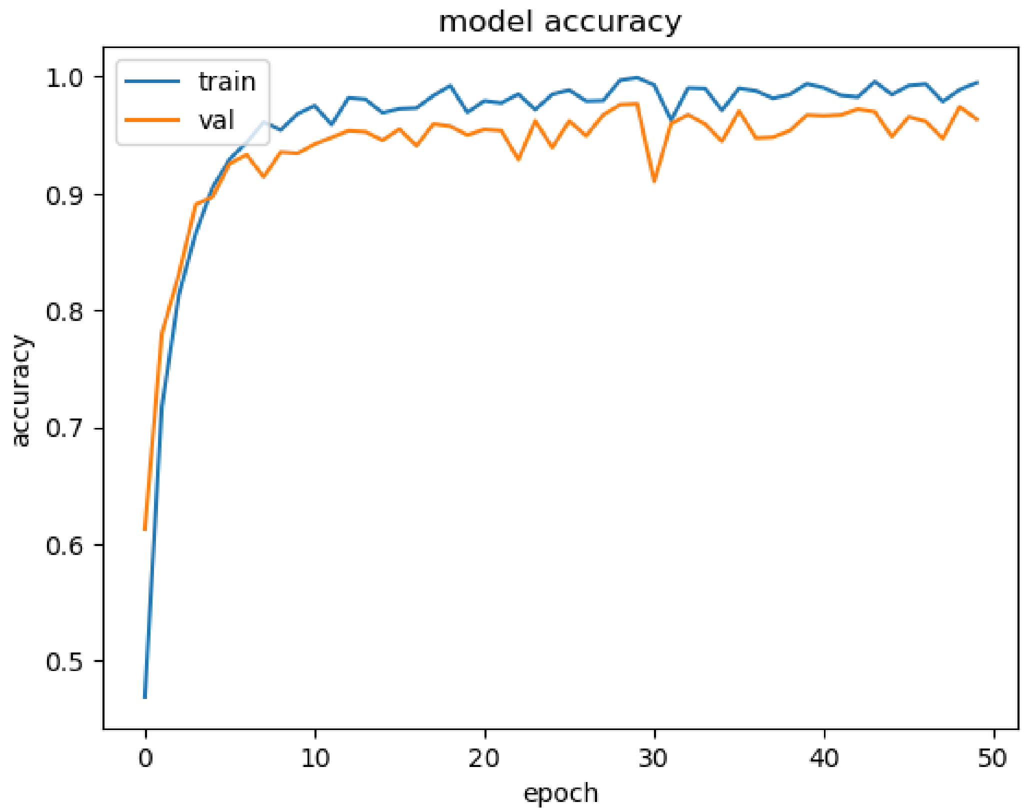
```
Epoch 1/50
235/235 [==============================] - 27s 110ms/step - loss: 2.0206 - accuracy: 0.46
92 - val_loss: 1.0094 - val_accuracy: 0.6126
Epoch 2/50
235/235 [==============================] - 25s 106ms/step - loss: 0.7880 - accuracy: 0.71
75 - val_loss: 0.6307 - val_accuracy: 0.7802
Epoch 3/50
235/235 [==============================] - 25s 104ms/step - loss: 0.5376 - accuracy: 0.81
29 - val_loss: 0.4791 - val_accuracy: 0.8308
Epoch 4/50
235/235 [==============================] - 25s 107ms/step - loss: 0.3945 - accuracy: 0.86
62 - val_loss: 0.3350 - val_accuracy: 0.8907
Epoch 5/50
235/235 [==============================] - 25s 107ms/step - loss: 0.2857 - accuracy: 0.90
54 - val_loss: 0.3125 - val_accuracy: 0.8973
Epoch 6/50
235/235 [==============================] - 25s 107ms/step - loss: 0.2118 - accuracy: 0.92
94 - val_loss: 0.2361 - val_accuracy: 0.9257
Epoch 7/50
235/235 [==============================] - 25s 106ms/step - loss: 0.1731 - accuracy: 0.94
37 - val_loss: 0.2292 - val_accuracy: 0.9330
Epoch 8/50
235/235 [==============================] - 25s 106ms/step - loss: 0.1251 - accuracy: 0.96
11 - val_loss: 0.2561 - val_accuracy: 0.9142
Epoch 9/50
235/235 [==============================] - 25s 107ms/step - loss: 0.1363 - accuracy: 0.95
43 - val_loss: 0.2284 - val_accuracy: 0.9354
Epoch 10/50
235/235 [==============================] - 25s 107ms/step - loss: 0.0989 - accuracy: 0.96
81 - val_loss: 0.2244 - val_accuracy: 0.9344
Epoch 11/50
235/235 [==============================] - 25s 106ms/step - loss: 0.0752 - accuracy: 0.97
53 - val_loss: 0.2408 - val_accuracy: 0.9422
Epoch 12/50
235/235 [==============================] - 25s 105ms/step - loss: 0.1162 - accuracy: 0.95
89 - val_loss: 0.2025 - val_accuracy: 0.9479
Epoch 13/50
235/235 [==============================] - 25s 106ms/step - loss: 0.0566 - accuracy: 0.98
20 - val_loss: 0.1957 - val_accuracy: 0.9537
Epoch 14/50
235/235 [==============================] - 25s 106ms/step - loss: 0.0574 - accuracy: 0.98
06 - val_loss: 0.2234 - val_accuracy: 0.9527
Epoch 15/50
235/235 [==============================] - 25s 105ms/step - loss: 0.0902 - accuracy: 0.96
90 - val_loss: 0.2037 - val_accuracy: 0.9455
Epoch 16/50
235/235 [==============================] - 25s 106ms/step - loss: 0.0811 - accuracy: 0.97
25 - val_loss: 0.1920 - val_accuracy: 0.9551
Epoch 17/50
235/235 [==============================] - 25s 104ms/step - loss: 0.0778 - accuracy: 0.97
32 - val_loss: 0.2300 - val_accuracy: 0.9407
Epoch 18/50
235/235 [==============================] - 25s 106ms/step - loss: 0.0488 - accuracy: 0.98
38 - val_loss: 0.1655 - val_accuracy: 0.9595
Epoch 19/50
235/235 [==============================] - 25s 105ms/step - loss: 0.0241 - accuracy: 0.99
22 - val_loss: 0.1860 - val_accuracy: 0.9575
Epoch 20/50
235/235 [==============================] - 25s 106ms/step - loss: 0.0955 - accuracy: 0.96
94 - val_loss: 0.2461 - val_accuracy: 0.9499
Epoch 21/50
235/235 [==============================] - 25s 106ms/step - loss: 0.0619 - accuracy: 0.97
93 - val_loss: 0.2351 - val_accuracy: 0.9549
Epoch 22/50
```

```
235/235 [==============================] - 25s 105ms/step - loss: 0.0671 - accuracy: 0.97
75 - val_loss: 0.2050 - val_accuracy: 0.9538
Epoch 23/50
235/235 [==============================] - 25s 105ms/step - loss: 0.0426 - accuracy: 0.98
51 - val_loss: 0.3018 - val_accuracy: 0.9290
Epoch 24/50
235/235 [==============================] - 25s 106ms/step - loss: 0.0855 - accuracy: 0.97
18 - val_loss: 0.2202 - val_accuracy: 0.9621
Epoch 25/50
235/235 [==============================] - 24s 103ms/step - loss: 0.0448 - accuracy: 0.98
49 - val_loss: 0.2810 - val_accuracy: 0.9393
Epoch 26/50
235/235 [==============================] - 25s 105ms/step - loss: 0.0354 - accuracy: 0.98
84 - val_loss: 0.2186 - val_accuracy: 0.9621
Epoch 27/50
235/235 [==============================] - 25s 105ms/step - loss: 0.0650 - accuracy: 0.97
91 - val_loss: 0.2246 - val_accuracy: 0.9493
Epoch 28/50
235/235 [==============================] - 25s 105ms/step - loss: 0.0641 - accuracy: 0.97
95 - val_loss: 0.2096 - val_accuracy: 0.9672
Epoch 29/50
235/235 [==============================] - 25s 106ms/step - loss: 0.0103 - accuracy: 0.99
70 - val_loss: 0.1647 - val_accuracy: 0.9760
Epoch 30/50
235/235 [==============================] - 25s 105ms/step - loss: 0.0028 - accuracy: 0.99
93 - val_loss: 0.1677 - val_accuracy: 0.9767
Epoch 31/50
235/235 [==============================] - 25s 106ms/step - loss: 0.0232 - accuracy: 0.99
28 - val_loss: 0.4777 - val_accuracy: 0.9101
Epoch 32/50
235/235 [==============================] - 25s 106ms/step - loss: 0.1247 - accuracy: 0.96
28 - val_loss: 0.2033 - val_accuracy: 0.9598
Epoch 33/50
235/235 [==============================] - 25s 107ms/step - loss: 0.0296 - accuracy: 0.99
03 - val_loss: 0.1893 - val_accuracy: 0.9674
Epoch 34/50
235/235 [==============================] - 25s 105ms/step - loss: 0.0299 - accuracy: 0.98
98 - val_loss: 0.2182 - val_accuracy: 0.9594
Epoch 35/50
235/235 [==============================] - 25s 106ms/step - loss: 0.0925 - accuracy: 0.97
12 - val_loss: 0.2705 - val_accuracy: 0.9446
Epoch 36/50
235/235 [==============================] - 25s 106ms/step - loss: 0.0307 - accuracy: 0.99
01 - val_loss: 0.1762 - val_accuracy: 0.9706
Epoch 37/50
235/235 [==============================] - 26s 109ms/step - loss: 0.0393 - accuracy: 0.98
78 - val_loss: 0.2368 - val_accuracy: 0.9475
Epoch 38/50
235/235 [==============================] - 25s 106ms/step - loss: 0.0584 - accuracy: 0.98
15 - val_loss: 0.2727 - val_accuracy: 0.9482
Epoch 39/50
235/235 [==============================] - 24s 103ms/step - loss: 0.0471 - accuracy: 0.98
48 - val_loss: 0.2883 - val_accuracy: 0.9538
Epoch 40/50
235/235 [==============================] - 25s 106ms/step - loss: 0.0197 - accuracy: 0.99
39 - val_loss: 0.1856 - val_accuracy: 0.9671
Epoch 41/50
235/235 [==============================] - 25s 106ms/step - loss: 0.0330 - accuracy: 0.99
04 - val_loss: 0.1897 - val_accuracy: 0.9664
Epoch 42/50
235/235 [==============================] - 25s 106ms/step - loss: 0.0528 - accuracy: 0.98
40 - val_loss: 0.2031 - val_accuracy: 0.9672
Epoch 43/50
235/235 [==============================] - 24s 104ms/step - loss: 0.0550 - accuracy: 0.98
25 - val_loss: 0.1851 - val_accuracy: 0.9722
Epoch 44/50
235/235 [==============================] - 25s 106ms/step - loss: 0.0123 - accuracy: 0.99
58 - val_loss: 0.2126 - val_accuracy: 0.9699
Epoch 45/50
235/235 [==============================] - 25s 105ms/step - loss: 0.0544 - accuracy: 0.98
46 - val_loss: 0.2614 - val_accuracy: 0.9486
Epoch 46/50
```

```
235/235 [==============================] - 25s 105ms/step - loss: 0.0225 - accuracy: 0.99
23 - val_loss: 0.2433 - val_accuracy: 0.9656
Epoch 47/50
235/235 [==============================] - 24s 104ms/step - loss: 0.0196 - accuracy: 0.99
37 - val_loss: 0.2668 - val_accuracy: 0.9617
Epoch 48/50
235/235 [==============================] - 25s 105ms/step - loss: 0.0704 - accuracy: 0.97
87 - val_loss: 0.2537 - val_accuracy: 0.9471
Epoch 49/50
235/235 [==============================] - 25s 106ms/step - loss: 0.0358 - accuracy: 0.98
88 - val_loss: 0.1756 - val_accuracy: 0.9743
Epoch 50/50
235/235 [==============================] - 25s 105ms/step - loss: 0.0167 - accuracy: 0.99
48 - val_loss: 0.2632 - val_accuracy: 0.9635
```

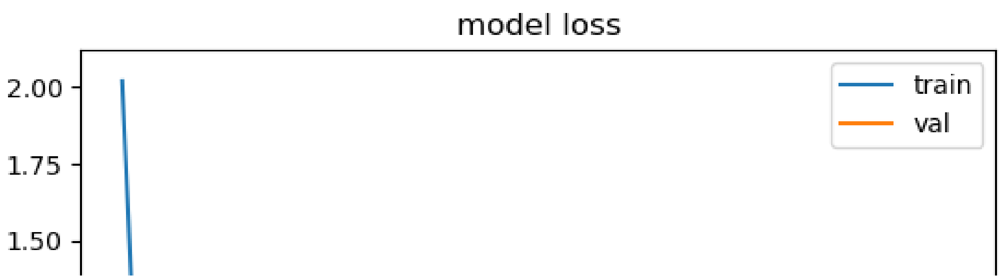
In [82]:

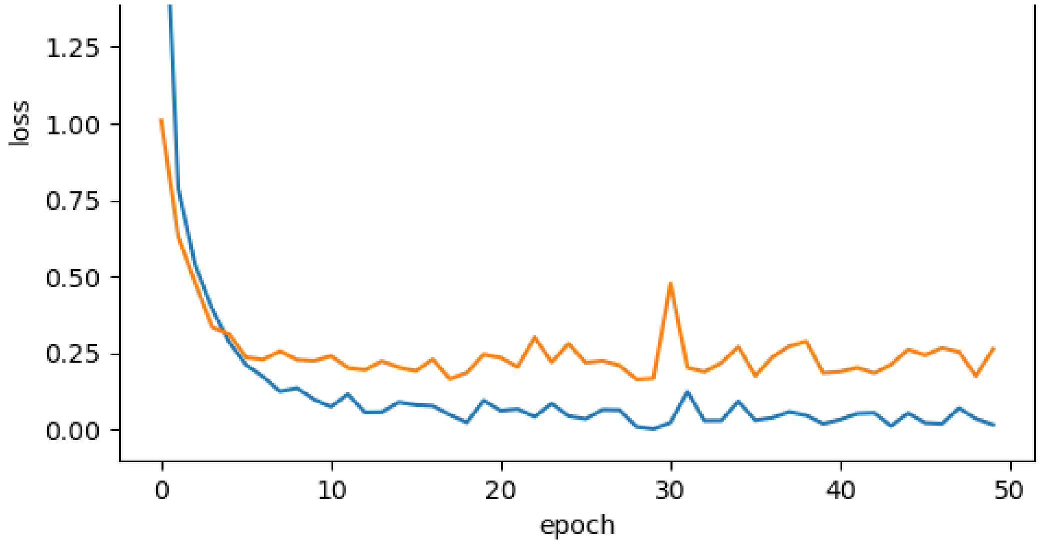
```python
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```



In [83]:

```python
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper right')
plt.show()
```





In [96]:

```
results = model.evaluate(X_test , Y_test, verbose=0)

print("    Test Loss: {:.5f}".format(results[0]))
print("Test Accuracy: {:.2f}%".format(results[1] * 100))
```

```
    Test Loss: 0.24558
Test Accuracy: 96.27%
```

In [97]:

```
from sklearn.metrics import confusion_matrix , classification_report

y_true = list(Y_test)
y_pred = model.predict(X_test)
y_pred = list(map(lambda x: np.argmax(x), y_pred))
print('Y Actual Values :' , y_true[0:10])
print('Y Predicted Values :' , y_pred[0:10])
```

```
294/294 [==============================] - 3s 9ms/step
Y Actual Values : [5, 1, 4, 0, 5, 0, 2, 0, 3, 2]
Y Predicted Values : [5, 1, 4, 0, 5, 0, 2, 0, 3, 2]
```

In [127]:

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

classes_labels=[]
for key in classes.keys():
    classes_labels.append(key)
print(classes_labels)
```

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

In [136]:

```
classes = {4: ('nv', ' melanocytic nevi'),
           6: ('mel', 'melanoma'),
           2 :('bkl', 'benign keratosis-like lesions'),
           1:('bcc' , ' basal cell carcinoma'),
           5: ('vasc', ' pyogenic granulomas and hemorrhage'),
           0: ('akiec', 'Actinic keratoses and intraepithelial carcinomae'),
           3: ('df', 'dermatofibroma')}
```

In [100]:

```
cm = confusion_matrix(y_true,y_pred,labels=classes_labels)
print(confusion_matrix(y_true,y_pred,labels=classes_labels))
```

```
[[1235    9    0   15    2    0    1]
 [  95 1079    2  161    7   25    5]
 [   0    0 1351    0    0    0    0]
```

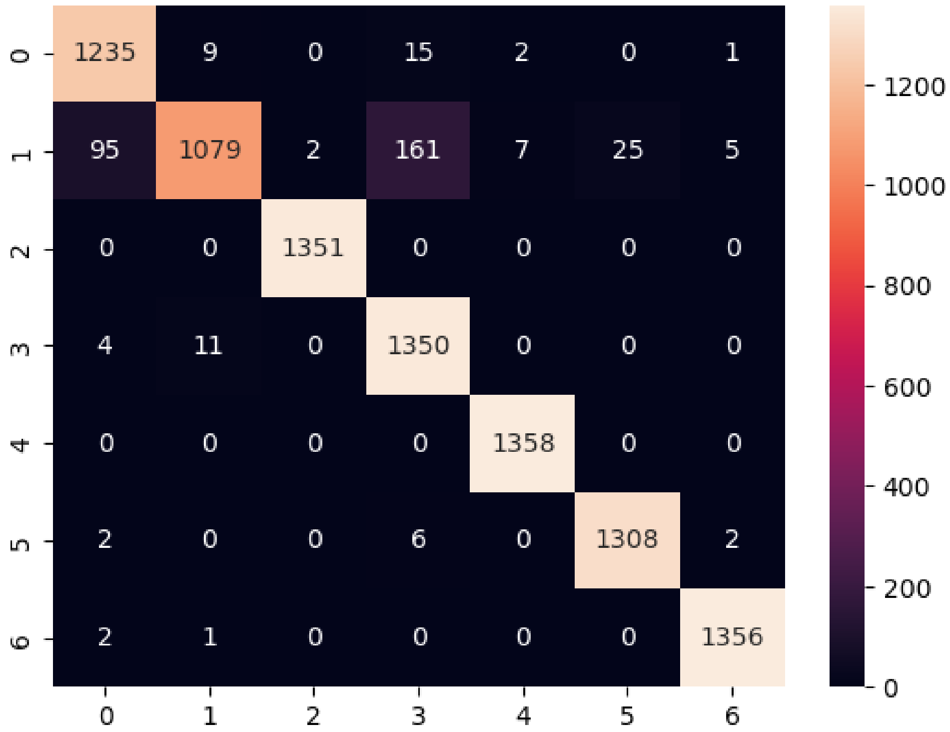
```
[   4   11    0 1350    0    0    0]
[   0    0    0    0 1358    0    0]
[   2    0    0    6    0 1308    2]
[   2    1    0    0    0    0 1356]]
```

In [101]:

```
sns.heatmap(cm, annot = True, fmt='')
```

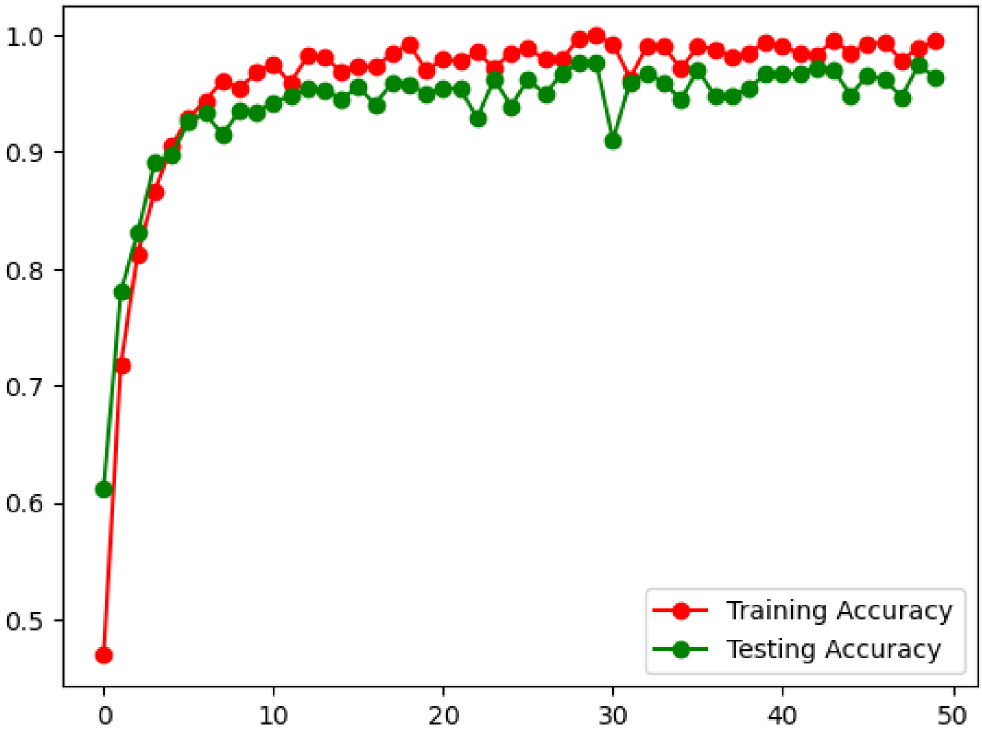
Out[101]:

```
<AxesSubplot:>
```



In [102]:

```
#training acc vs testing acc graph
plt.plot(history.history["accuracy"] , 'ro-' , label = "Training Accuracy")
plt.plot(history.history["val_accuracy"] , 'go-' , label = "Testing Accuracy")
plt.legend()
plt.show()
```



In [137]:

```
#predicting
y_pred  = model.predict(X_test).round()
```

```
294/294 [==============================] - 3s 10ms/step
```

In [138]:

```
target_names = [f"{classes[i]}" for i in range(7)]
print(len(Y_test) ,"  ",len(y_pred))
y_pred = list(map(lambda x: np.argmax(x), y_pred))
print(classification_report(Y_test , y_pred,target_names=target_names))
```

```
9387    9387
                                                              precision    recall  f1-
score   support

('akiec', 'Actinic keratoses and intraepithelial carcinomae')       0.99      1.00      0
.99      1359
                              ('bcc', ' basal cell carcinoma')       0.98      0.99
0.99      1318
                      ('bkl', 'benign keratosis-like lesions')       0.92      0.98
0.95      1262
                                      ('df', 'dermatofibroma')       1.00      1.00
1.00      1351
                                   ('nv', ' melanocytic nevi')       0.98      0.79
0.87      1374
               ('vasc', ' pyogenic granulomas and hemorrhage')       0.99      1.00
1.00      1358
                                           ('mel', 'melanoma')       0.88      0.99
0.93      1365

                                                    accuracy
0.96      9387
                                                   macro avg       0.96      0.96
0.96      9387
                                                weighted avg       0.96      0.96
0.96      9387
```

In [ ]:

```

```