Skin Disease Classification

```
from fastai import *
from fastai.vision import *
from fastai.callbacks.hooks import *

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import auc,roc_curve

import os
print(os.listdir("../input"))
```

```
['hmnist_28_28_RGB.csv', 'hmnist_28_28_L.csv', 'ham10000_images_part_2', 'ham1
0000_images_part_1', 'hmnist_8_8_RGB.csv', 'HAM10000_metadata.csv', 'hmnist_8_
8_L.csv']
```

Exploratory Data Analysis

In [2]:

```
# Paths and roots to the important files
path='../input/'
csv_file='../input/HAM10000_metadata.csv'
```

In [3]:

```
df=pd.read_csv(csv_file).set_index('image_id')
df.head()
```

ut[3]:

In [4]:

```
# Categories of the diferent diseases
lesion_type_dict = {
    'nv': 'Melanocytic nevi',
    'mel': 'Melanoma',
    'bkl': 'Benign keratosis ',
    'bcc': 'Basal cell carcinoma',
    'akiec': 'Actinic keratoses',
    'vasc': 'Vascular lesions',
    'df': 'Dermatofibroma'
}
```

In [5]:

```
df.dx=df.dx.astype('category',copy=True)
df['labels']=df.dx.cat.codes # Convert the labels to numbers
df['lesion']= df.dx.map(lesion_type_dict)
df.head()
```

Out[5]:

In [6]:

```
print(df.lesion.value_counts())
```

```
Melanocytic nevi        6705
Melanoma                1113
Benign keratosis        1099
```

```
Basal cell carcinoma     514
Actinic keratoses        327
Vascular lesions         142
Dermatofibroma           115
Name: lesion, dtype: int64
```

In [7]:

```
df.loc['ISIC_0027419','lesion']
```

Out[7]:

```
'Benign keratosis '
```

Countplot

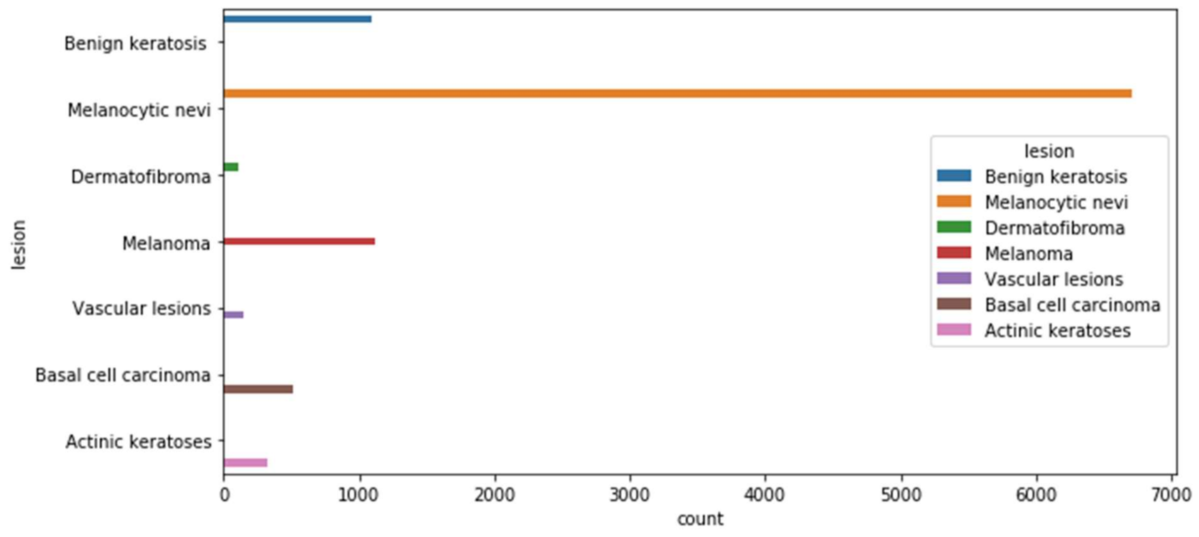
Here we notice tha we have data imbalance

In [8]:

```
fig, ax1 = plt.subplots(1, 1, figsize = (10, 5))
sns.countplot(y='lesion',data=df, hue="lesion",ax=ax1)
```

Out[8]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff27a102550>
```



Dataset

In [9]:

```
class CustomImageItemList(ImageItemList):
    def custom_label(self,df, **kwargs)->'LabelList':
        """Custom Labels from path"""
        file_names=np.vectorize(lambda files: str(files).split('/')[-1][:-4])
        get_labels=lambda x: df.loc[x,'lesion']
        #self.items is an np array of PosixPath objects with each image path
        labels= get_labels(file_names(self.items))
        y = CategoryList(items=labels)
        res = self._label_list(x=self,y=y)
        return res
```

In [10]:

```
def get_data(bs, size):
    train_ds = (CustomImageItemList.from_folder('../input', extensions='.jpg')
                    .random_split_by_pct(0.15)
                    .custom_label(df)
                    .transform(tfms=get_transforms(flip_vert=True),size=size)
                    .databunch(num_workers=2, bs=bs)
```

```
                        .normalize(imagenet_stats))
    return train_ds
```

In [11]:

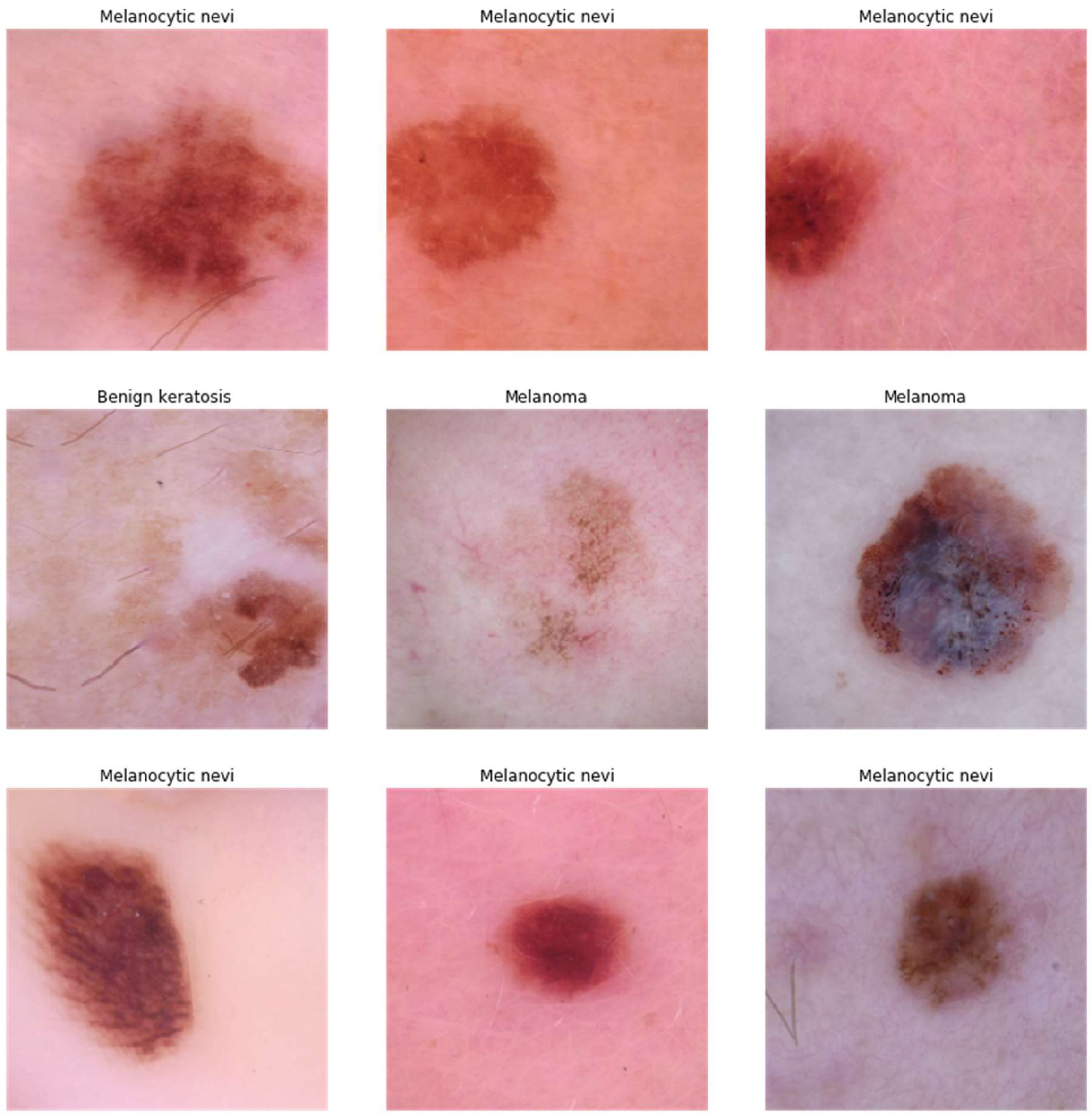
```
data=get_data(16,224)
```

In [12]:

```
data.classes=list(np.unique(df.lesion))
data.c= len(np.unique(df.lesion))
```

In [13]:

```
data.show_batch(rows=3)
```



Model ResNet50

In [14]:

```
learner=create_cnn(data,models.resnet50,metrics=[accuracy], model_dir="/tmp/model/
")
```

```
Downloading: "https://download.pytorch.org/models/resnet50-19c8e357.pth" to /t
mp/.torch/models/resnet50-19c8e357.pth
100%|██████████| 102502400/102502400 [00:03<00:00, 26653549.20it/s]
```

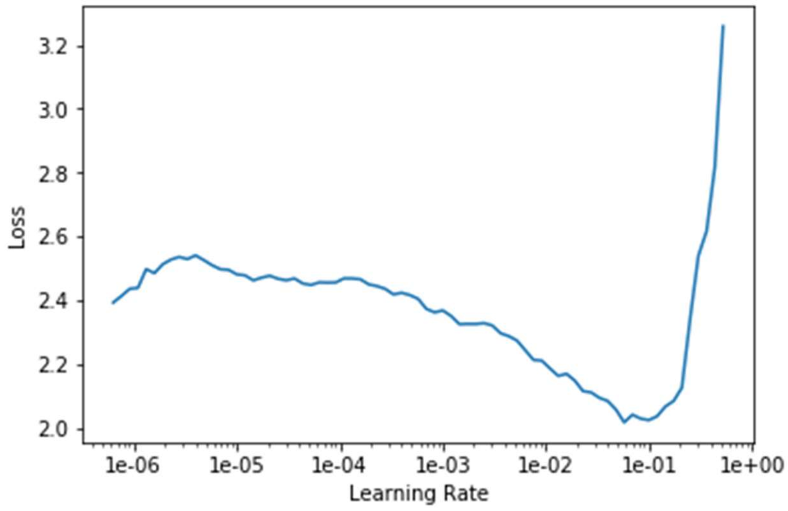
In [15]:

```
learner.loss_func=nn.CrossEntropyLoss()
```

In [16]:

```
learner.lr_find()
learner.recorder.plot()
```

```
LR Finder is complete, type {learner_name}.recorder.plot() to see the graph.
```



In [17]:

```
learner.fit_one_cycle(10, 3e-3)
```

60.00% [6/10 31:58<21:19]

epoch	train_loss	valid_loss	accuracy
1	0.841194	0.700374	0.766977
2	0.737256	0.625305	0.772304

3	0.738680	0.600183	0.790945
4	0.624489	0.623843	0.771638
5	0.575161	0.586372	0.795606
6	0.476066	0.502809	0.822903

```
97.74% [520/532 04:32<00:06 0.5225]
```

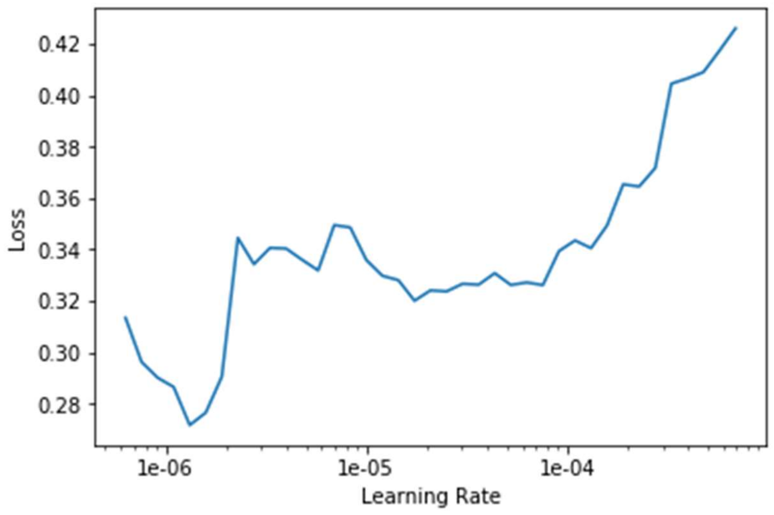
In [18]:

```
learner.unfreeze()
```

In [19]:

```
learner.lr_find()
learner.recorder.plot()
```

```
LR Finder is complete, type {learner_name}.recorder.plot() to see the graph.
```



In [20]:

```
lr=1e-6
learner.fit_one_cycle(3, slice(3*lr,10*lr))
```

Total time: 16:47

epoch	train_loss	valid_loss	accuracy
1	0.353880	0.407866	0.846871
2	0.398067	0.393029	0.851531
3	0.374383	0.395229	0.8521

In [21]:

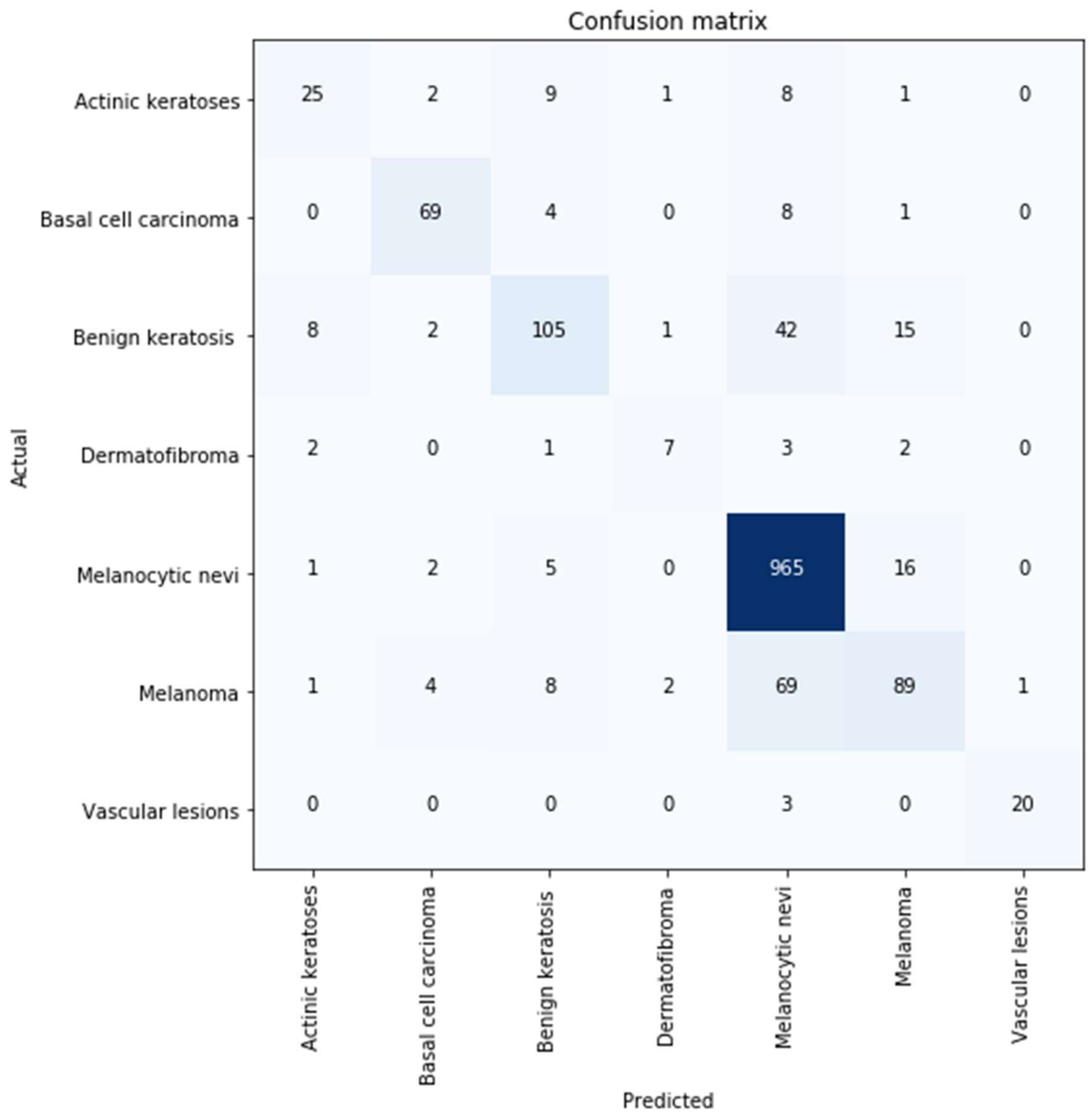
```
learner.save('stage-1')
```

In [22]:

```
interp = ClassificationInterpretation.from_learner(learner)
```

In [23]:

```
interp.plot_confusion_matrix(figsize=(10,8))
```



	Actinic keratoses	Basal cell carcinoma	Benign keratosis	Dermatofibroma	Melanocytic nevi	Melanoma	Vascular lesions
Actinic keratoses	25	2	9	1	8	1	0
Basal cell carcinoma	0	69	4	0	8	1	0
Benign keratosis	8	2	105	1	42	15	0
Dermatofibroma	2	0	1	7	3	2	0
Melanocytic nevi	1	2	5	0	965	16	0
Melanoma	1	4	8	2	69	89	1
Vascular lesions	0	0	0	0	3	0	20

In [24]:

```
interp.most_confused()
```

Out[24]:

```
[('Melanoma', 'Melanocytic nevi', 69),
 ('Benign keratosis ', 'Melanocytic nevi', 42),
 ('Melanocytic nevi', 'Melanoma', 16),
 ('Benign keratosis ', 'Melanoma', 15),
 ('Actinic keratoses', 'Benign keratosis ', 9),
 ('Actinic keratoses', 'Melanocytic nevi', 8),
 ('Basal cell carcinoma', 'Melanocytic nevi', 8),
 ('Benign keratosis ', 'Actinic keratoses', 8),
 ('Melanoma', 'Benign keratosis ', 8),
 ('Melanocytic nevi', 'Benign keratosis ', 5),
 ('Basal cell carcinoma', 'Benign keratosis ', 4),
 ('Melanoma', 'Basal cell carcinoma', 4),
 ('Dermatofibroma', 'Melanocytic nevi', 3),
 ('Vascular lesions', 'Melanocytic nevi', 3),
 ('Actinic keratoses', 'Basal cell carcinoma', 2),
 ('Benign keratosis ', 'Basal cell carcinoma', 2),
 ('Dermatofibroma', 'Actinic keratoses', 2),
 ('Dermatofibroma', 'Melanoma', 2),
 ('Melanocytic nevi', 'Basal cell carcinoma', 2),
 ('Melanoma', 'Dermatofibroma', 2)]
```

Inference

In [25]:

```
pred_data=get_data(16,224)
```

In [26]:

```
pred_data.classes=list(np.unique(df.lesion))
pred_data.c= len(np.unique(df.lesion))
```

In [27]:

```
pred_data.single_from_classes(path, pred_data.classes)
```

```
/opt/conda/lib/python3.6/site-packages/fastai/data_block.py:388: UserWarning:
Your training set is empty. Is this is by design, pass `ignore_empty=True` to
remove this warning.
  warn("Your training set is empty. Is this is by design, pass `ignore_empty=T
rue` to remove this warning.")
/opt/conda/lib/python3.6/site-packages/fastai/data_block.py:391: UserWarning:
Your validation set is empty. Is this is by design, use `no_split()`
                 or pass `ignore_empty=True` when labelling to remove this war
ning.
  or pass `ignore_empty=True` when labelling to remove this warning.""")
```

Out[27]:

```
ImageDataBunch;

Train: LabelList
y: CategoryList (0 items)
[]...
Path: ../input
x: ImageItemList (0 items)
[]...
Path: ../input;

Valid: LabelList
y: CategoryList (0 items)
[]...
Path: ../input
x: ImageItemList (0 items)
[]...
Path: ../input;

Test: None
```

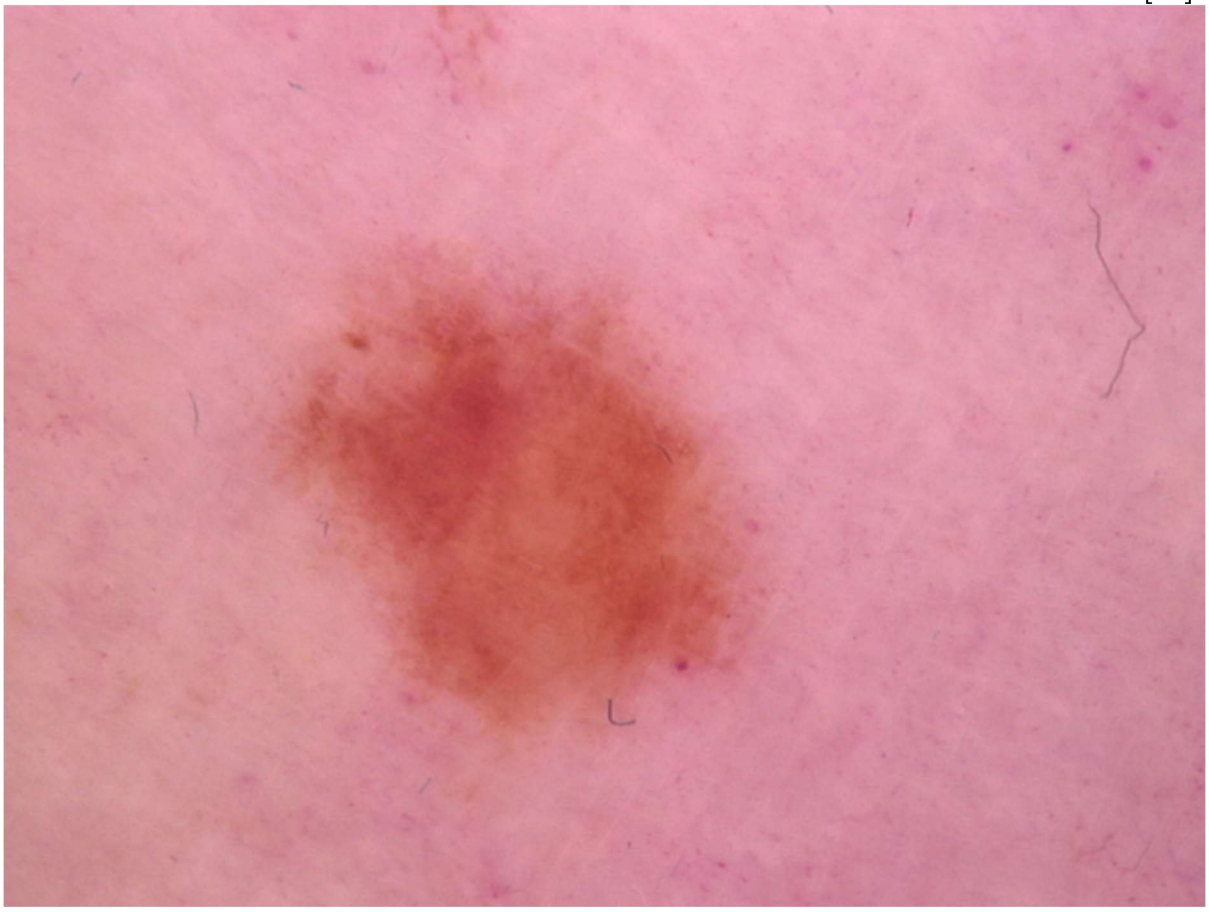
In [28]:

```
predictor = create_cnn(pred_data, models.resnet50, model_dir="/tmp/model/").load('
stage-1')
```

In [29]:

```
img = open_image('../input/ham10000_images_part_2/ISIC_0029886.jpg')
img
```

Out[29]:



In [30]:

```
pred_class,pred_idx,outputs = predictor.predict(img)
pred_class
```

Out[30]:

```
Category Melanocytic nevi
```

Predictions

In [31]:

```
# Predictions of the validation data
preds_val, y_val=learner.get_preds()
```

Roc Curve

With the ROC curve we will mesuare how good it's our model

In [32]:

```
#  ROC curve
fpr, tpr, thresholds = roc_curve(y_val.numpy(), preds_val.numpy()[:,1], pos_label=
1)

#  ROC area
pred_score = auc(fpr, tpr)
```

```
print(f'ROC area is {pred_score}')
```

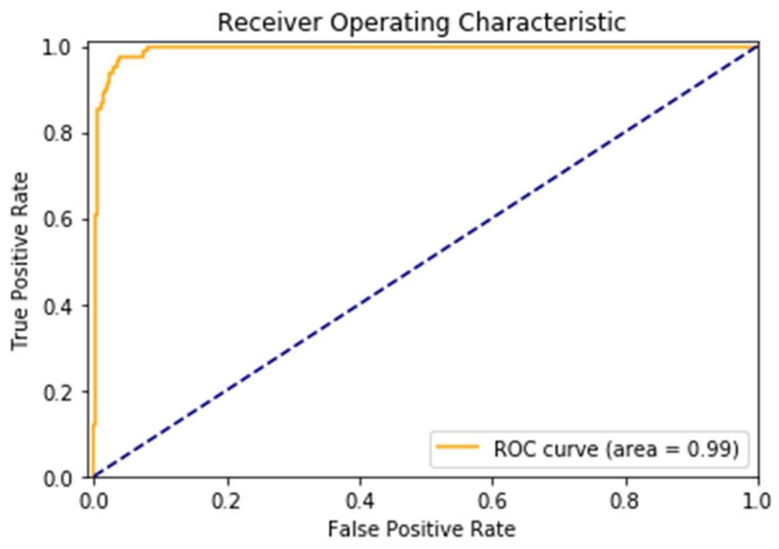
```
ROC area is 0.9935846788045345
```

In [33]:

```
plt.figure()
plt.plot(fpr, tpr, color='orange', label='ROC curve (area = %0.2f)' % pred_score)
plt.plot([0, 1], [0, 1], color='navy', linestyle='--')
plt.xlim([-0.01, 1.0])
plt.ylim([0.0, 1.01])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
```

Out[33]:

```
<matplotlib.legend.Legend at 0x7ff248dac860>
```



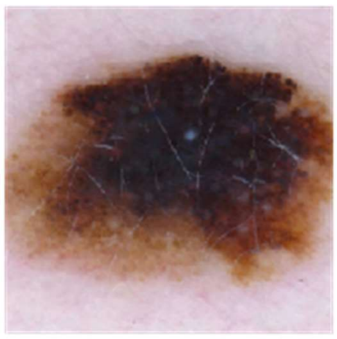
Heatmap

In [34]:

```
x,y = data.valid_ds[2]
x.show()
data.valid_ds.y[2]
```

Out[34]:

```
Category Melanocytic nevi
```



In [35]:

```
def heatMap(x,y,data, learner, size=(0,224,224,0)):
    """HeatMap"""
    # Evaluation mode
    m=learner.model.eval()

    # Denormalize the image
    xb,_ = data.one_item(x)
    xb_im = Image(data.denorm(xb)[0])
    xb = xb.cuda()

    # hook the activations
    with hook_output(m[0]) as hook_a:
        with hook_output(m[0], grad=True) as hook_g:
            preds = m(xb)
            preds[0,int(y)].backward()

    # Activations
    acts=hook_a.stored[0].cpu()

    # Avg of the activations
    avg_acts=acts.mean(0)

    # Show HeatMap
    _,ax = plt.subplots()
    xb_im.show(ax)
    ax.imshow(avg_acts, alpha=0.6, extent=size,
              interpolation='bilinear', cmap='magma')
```

In [36]:

```
heatMap(x,y,pred_data,learner)
```