1. Importing Required Libraries

In [6]:
```
# Import Required Python Packages :

import warnings
warnings.filterwarnings('ignore')

# Setting up our enviroment
# Data Viz & Regular Expression Libraries :
%reload_ext autoreload
%autoreload 2
%matplotlib inline

# Scientific and Data Manipulation Libraries :
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Import FastAI library
from fastai import *
from fastai.vision import *
from fastai.metrics import error_rate
import os
```

In [7]:
```
x  = '/kaggle/input/cat-and-dog/training_set/training_set'
path = Path(x)
path.ls()
```

Out[7]:
```
[PosixPath('/kaggle/input/cat-and-dog/training_set/training_set/dogs'),
 PosixPath('/kaggle/input/cat-and-dog/training_set/training_set/cats')]
```

In [8]:
```

np.random.seed(40)
data = ImageDataBunch.from_folder(path, train = '.', valid_pct=0.2,
                                  ds_tfms=get_transforms(), size=224,
                                  num_workers=4).normalize(imagenet_stats)
```

In [25]:
```
data.show_batch(rows=4, figsize=(7,6),recompute_scale_factor=True)
```

In [10]:
```
data
```

Out[10]:
```
ImageDataBunch;

Train: LabelList (6404 items)
x: ImageList
Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 22
4),Image (3, 224, 224)
y: CategoryList
dogs,dogs,dogs,dogs,dogs
Path: /kaggle/input/cat-and-dog/training_set/training_set;

Valid: LabelList (1601 items)
x: ImageList
Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 22
4),Image (3, 224, 224)
y: CategoryList
cats,dogs,cats,dogs,dogs
Path: /kaggle/input/cat-and-dog/training_set/training_set;

Test: None
```

In [11]:
```
print(data.classes)
len(data.classes)
data.c
```

```
['cats', 'dogs']
```

Out[11]: 2

Create Model

In [12]:
```
learn = cnn_learner(data, models.resnet34, metrics=[accuracy], model_dir = Pat
```

```
Downloading: "https://download.pytorch.org/models/resnet34-333f7ec4.pth" to /
root/.cache/torch/checkpoints/resnet34-333f7ec4.pth

HBox(children=(FloatProgress(value=0.0, max=87306240.0), HTML(value='')))
```

Training Neural Network

To find the perfect learning rates we can use the lr_find and recorder.plot methods which create a plot that relates the learning rate with the loss.

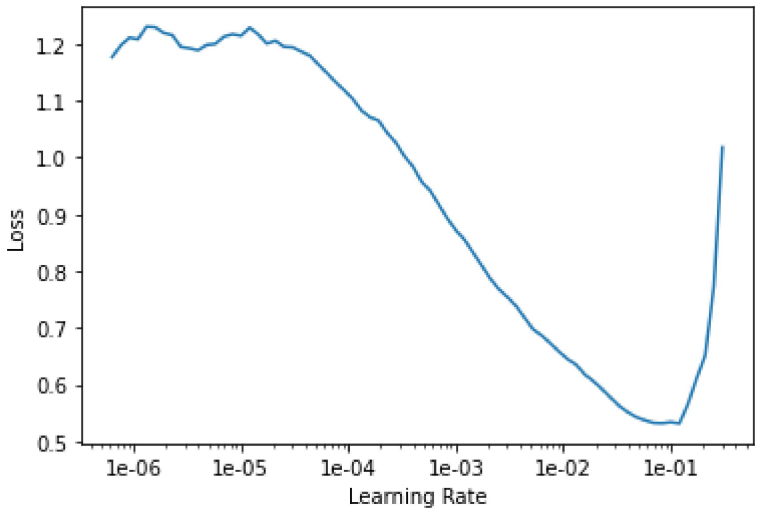
In [13]:
```
learn.lr_find()
learn.recorder.plot(suggestions=True)
```

0.00% [0/1 00:00<00:00]

epoch	train_loss	valid_loss	accuracy	time

86.00% [86/100 01:16<00:12 1.4306]

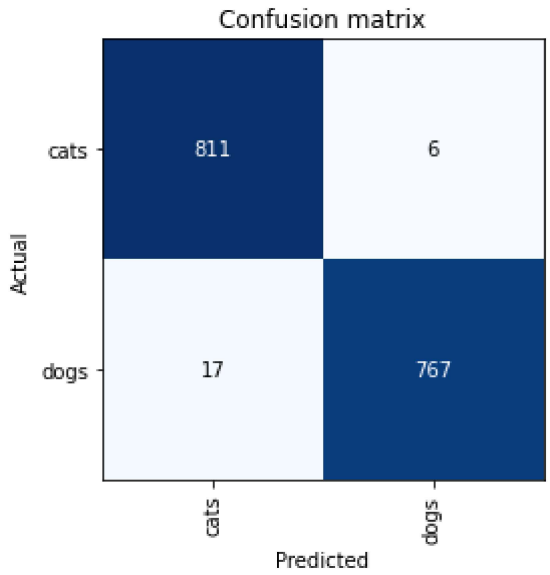
```
LR Finder is complete, type {learner_name}.recorder.plot() to see the graph.
```



In [14]:
```
lr1 = 1e-3
lr2 = 1e-1
learn.fit_one_cycle(4,slice(lr1,lr2))
```

epoch	train_loss	valid_loss	accuracy	time
0	0.548361	0.349702	0.968145	01:40
1	0.418156	0.132228	0.972517	01:39
2	0.137788	0.065364	0.978139	01:40
3	0.065808	0.046996	0.985634	01:40

In [15]:
```
interp = ClassificationInterpretation.from_learner(learn)
interp.plot_confusion_matrix()
```



5. Prediction using trained model

In [24]:
```
img = open_image('../input/cat-and-dog/test_set/test_set/dogs/dog.4001.jpg')
print(learn.predict(img)[0])
img
```

```
dogs
```

Out[24]:

In [ ]:
```

```