The notebook shows the classification of photos of real estate interiors and exteriors.

The classes are - bedrooms, bathrooms, front exterior, back yards, kitchen, dining rooms, living room.

The photos for training are downloaded from Google Images. After training, the model can be used to identify the room in photographs.

```
In [0]: #Import packages
        from fastai.vision import *
        import numpy as np
```

```
In [0]: #Put at beginning of every notebook to map Google drive to Colab
        from google.colab import drive
        drive.mount('/content/gdrive', force_remount=True)
        root_dir = "/content/gdrive/My Drive/"
        base_dir = root_dir + 'fastai-v3/'

        Mounted at /content/gdrive
```

Search for images in Google Images using the keywords and save the URLs in a file. Use script in Javscript console

urls = Array.from(document.querySelectorAll('.rg_di .rg_meta')).map(el=>JSON.parse(el.textContent).ou); window.open('data:text/csv;charset=utf-8,' + escape(urls.join('\n')));

Copy file into Google drive path and name it 'key word+ _urls'

Later we will download the images at these URLs and use them to train the model.

Download pictures and upload into appropriate directories in Google Drive

```
In [0]: labels = ['bedroom', 'bathroom', 'living', 'dining', 'kitchen', 'front', 'back yard']
        path = Path(base_dir + 'data/realestate')
```

Alternative method for downloading images.

I am using a Windows PC and sometimes the files containing URLs downloaded by running the script do not work, especially if you open them in Windows and save them. Hence the method below is better if you are using Windows.

Use Chrome extension for downloading images. Upload images into Google folder named as the classes and delete images which do not show as thumbnails or are not suitable.

Verify images and delete bad ones

In [0]:
```python
classes = ['bedroom', 'bathroom', 'living', 'dining', 'kitchen', 'front', 'backyard']
for c in classes:
    print(c)
    verify_images(path/c, delete=True, max_size=500)
```

```
bedroom

100.00% [1359/1359 03:29<00:00]

bathroom

100.00% [459/459 00:51<00:00]

living

100.00% [733/733 01:26<00:00]

dining

100.00% [1207/1207 02:21<00:00]

kitchen

100.00% [931/931 01:43<00:00]

front

100.00% [914/914 01:43<00:00]

backyard

100.00% [706/706 01:14<00:00]
```

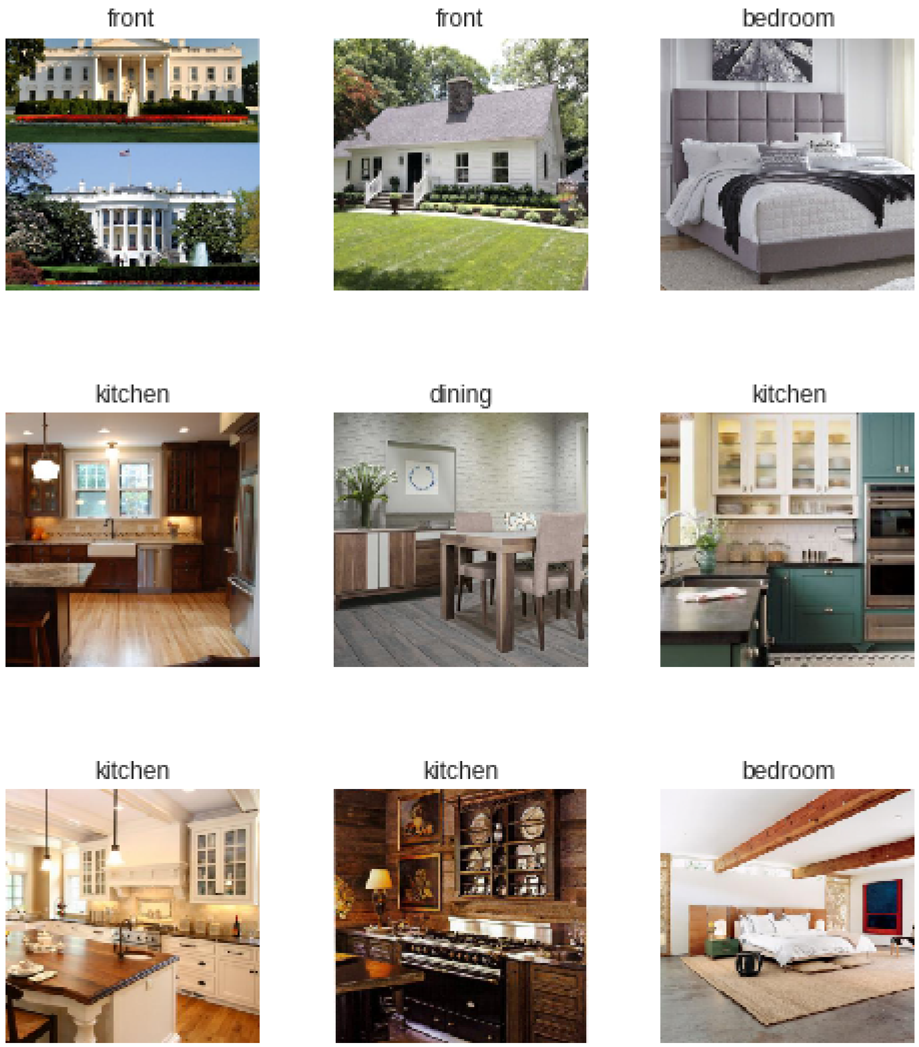
Create data object

In [0]:
```python
np.random.seed(123)
data = ImageDataBunch.from_folder(path, train=".", valid_pct=0.2, ds_tfms=get_transforms(do_flip=False, flip_vert=False, max_rotate=0.0, max_zoom=0.0, max_lighting=0.0,max_warp=0.0), size=224, num_workers=4).normalize(imagenet_stats)
```

In [0]:
```python
data.classes
```

Out[0]:
```
['backyard', 'bathroom', 'bedroom', 'dining', 'front', 'kitchen', 'living']
```

```
In [0]: data.show_batch(rows=3, figsize=(7,8))
```



Train the model using pre-trained model resnet34 and fit model

```
In [0]: learn = create_cnn(data, models.resnet34, metrics=error_rate)
```

```
In [0]: learn.fit_one_cycle(4)
```

Total time: 05:21

epoch	train_loss	valid_loss	error_rate
1	1.133326	0.665635	0.222839
2	0.761083	0.553283	0.182395
3	0.578217	0.518240	0.174465
4	0.496826	0.515675	0.176051

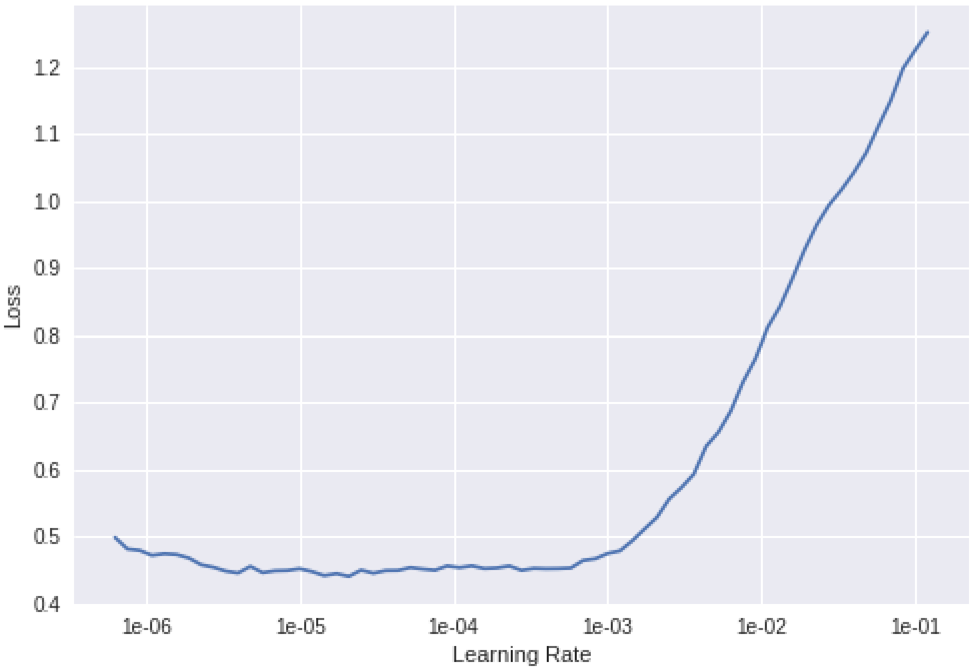
In [0]:
```
learn.unfreeze()
learn.lr_find()
learn.recorder.plot()
```

```
50.00% [1/2 01:07<01:07]

epoch   train_loss   valid_loss   error_rate
    1     1.296926

Interrupted

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



In [0]:
```
learn.fit_one_cycle(4, max_lr=7e-05)
```

```
Total time: 05:32

epoch   train_loss   valid_loss   error_rate
    1     0.442812     0.486324     0.167328
    2     0.359549     0.481119     0.161776
    3     0.228340     0.458382     0.153053
    4     0.150960     0.458080     0.148295
```

In [0]:
```
#Again create data with higher resolution
data = ImageDataBunch.from_folder(path, train=".", valid_pct=0.2, ds_tfms=get_
transforms(do_flip=False, flip_vert=False, max_rotate=0.0, max_zoom=0.0, max_l
ighting=0.0,max_warp=0.0), size=512, num_workers=4).normalize(imagenet_stats)
```

In [0]:
```
#Transfer learning
learn.data = data
```

In [0]:
```
learn.freeze()
```

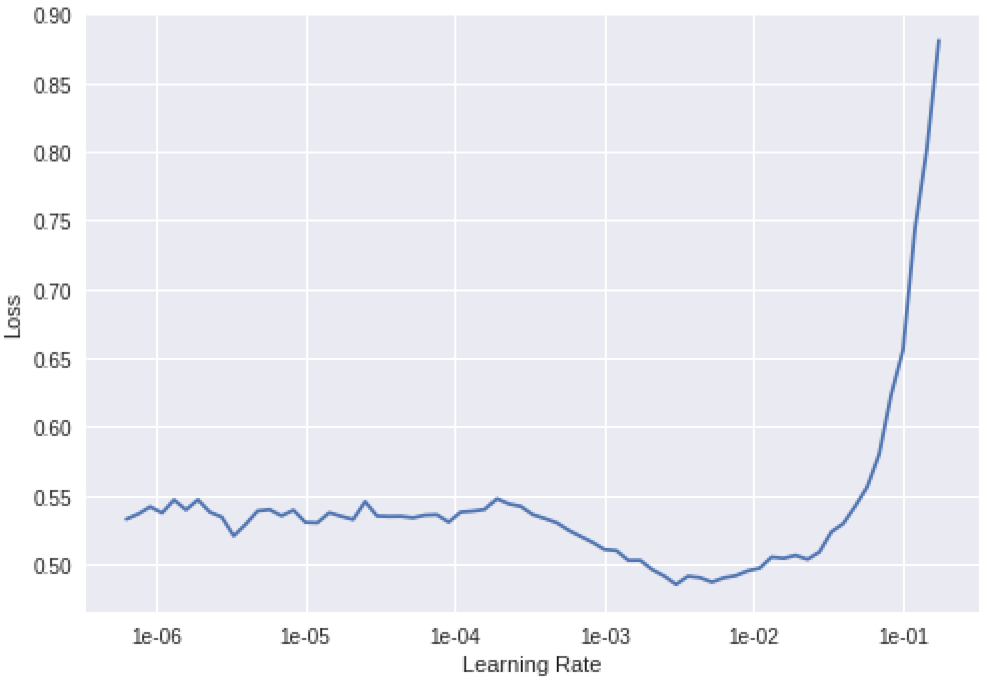
In [0]:
```
learn.lr_find()
learn.recorder.plot()
```

50.00% [1/2 03:09<03:09]

epoch	train_loss	valid_loss	error_rate
1	0.801979		

Interrupted

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



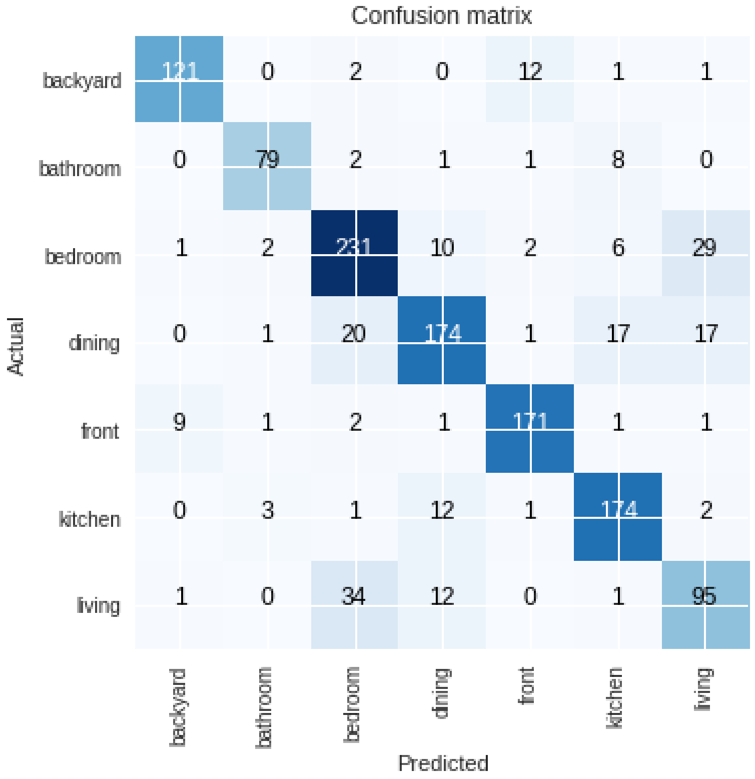
In [0]:
```
learn.fit_one_cycle(4, max_lr=5e-04)
```

Total time: 15:51

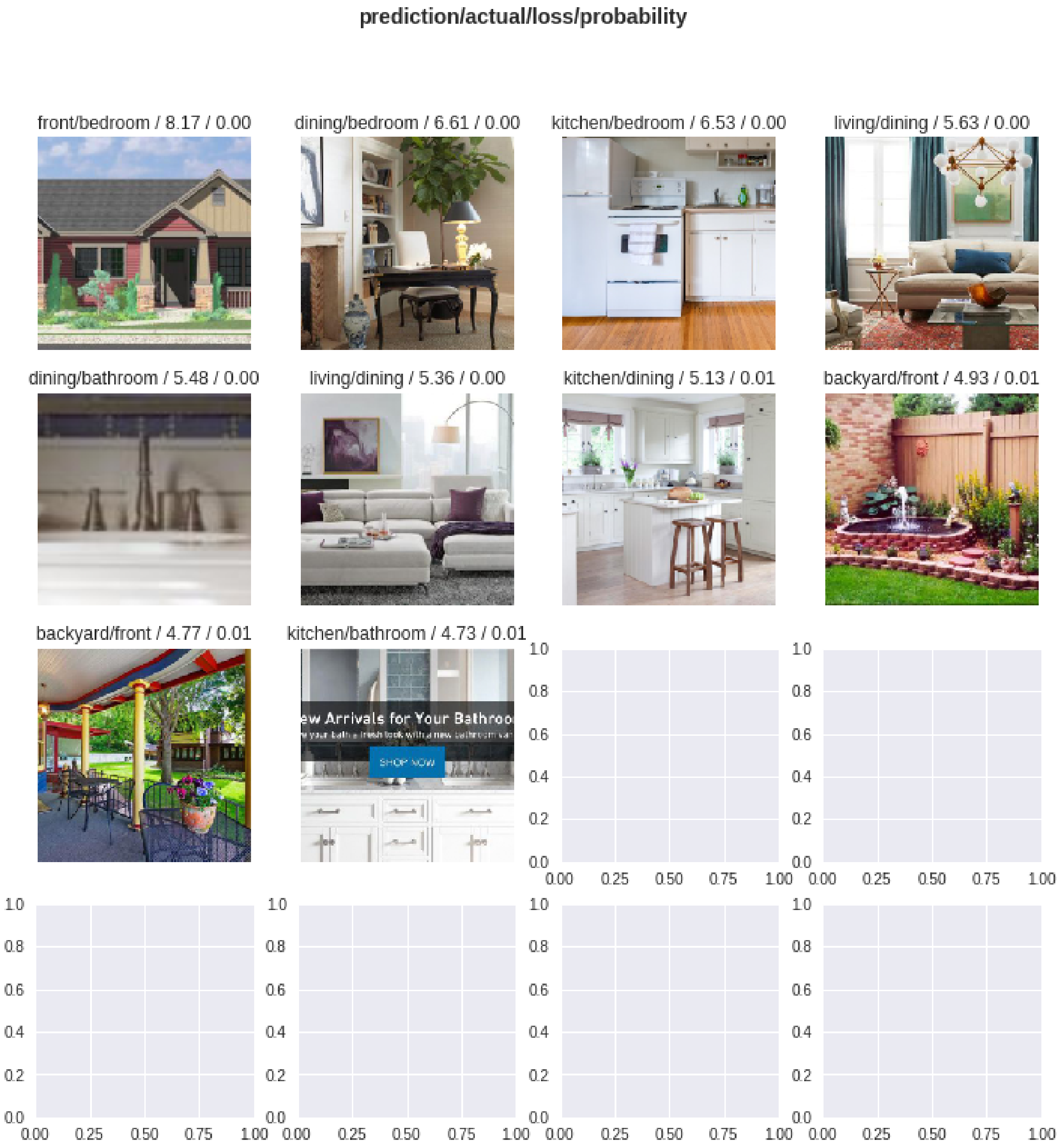
epoch	train_loss	valid_loss	error_rate
1	0.495526	0.343121	0.112609
2	0.419269	0.329292	0.104679
3	0.371824	0.327216	0.111023
4	0.326143	0.324021	0.107058

In [0]:
```
interp = ClassificationInterpretation.from_learner(learn)
```

```
In [0]: interp.plot_confusion_matrix()
```



```
In [0]: interp.plot_top_losses(10)
```



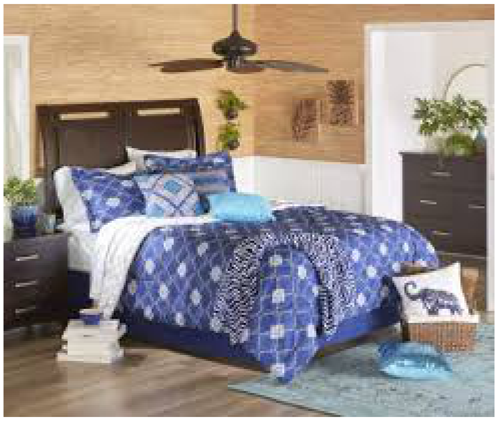
Export model for portability and do prediction

```
In [0]: learn.export()
```

In [0]:

```
# Impoort new image
img = open_image(path/'pic6.jpg')
img
```

Out[0]:



In [0]:

```
#Import exported model
learn = load_learner(path)
```

In [0]:

```
#Predict class of image
pred_class,pred_idx,outputs = learn.predict(img)
pred_class
```

Out[0]:

```
Category bedroom
```

The error rate for the above model is not great but considering that it was created by random images downloaded from Google, it is pretty good. It can be improved by cleaning up the training images and removing junk.