In [1]:
```
import os
from fastai import *
from fastai.vision import *
from fastai.callbacks import *
import numpy as np
import pandas as pd
pd.options.display.max_rows = 999
```

Modifying the the Test-Train CSV to work with fastai

In [2]:
```
df_Test_Train = pd.read_csv('cropped_test_train.csv')
df_Test_Train.drop(df_Test_Train.columns[df_Test_Train.columns.str.contains('unnamed',case = False)],axis = 1, inplace = True)
df_Test_Train = df_Test_Train.drop(columns=['index'])
df_Test_Train.head(100)
```

19	./Cropped_Egg_images/Clutch1_D18/egg34.JPG	1	18	34	Female
20	./Cropped_Egg_images/Clutch1_D18/egg35.JPG	1	18	35	Male
21	./Cropped_Egg_images/Clutch1_D18/egg36.JPG	1	18	36	Male
22	./Cropped_Egg_images/Clutch1_D18/egg37.JPG	1	18	37	Female
23	./Cropped_Egg_images/Clutch1_D18/egg38.JPG	1	18	38	Male
24	./Cropped_Egg_images/Clutch1_D18/egg39.JPG	1	18	39	Male
25	./Cropped_Egg_images/Clutch1_D18/egg40.JPG	1	18	40	Female
26	./Cropped_Egg_images/Clutch1_D18/egg41.JPG	1	18	41	Male
27	./Cropped_Egg_images/Clutch1_D18/egg42.JPG	1	18	42	Male
28	./Cropped_Egg_images/Clutch1_D18/egg45.JPG	1	18	45	Female
29	./Cropped_Egg_images/Clutch1_D18/egg46.JPG	1	18	46	Male
30	./Cropped_Egg_images/Clutch1_D18/egg47.JPG	1	18	47	Male
31	./Cropped_Egg_images/Clutch1_D18/egg51.JPG	1	18	51	Male

In [3]:
```
def dropdot(row):
    filepath = row['Crp_Filepath']
    newfilepath = filepath.split('./')[1]
    return newfilepath

df_Test_Train['Crp_Filepath'] = df_Test_Train.apply(dropdot, axis =1)

#Get just the D18
df_Test_Train = df_Test_Train[df_Test_Train['Day']==14]
df_Test_Train = df_Test_Train.reset_index()
df_Test_Train = df_Test_Train.drop(columns = ['index'])

df_Test_Train.head()

df_Test_Train.to_csv('Fastai_dataset_usable_d14.csv')
```

In [4]:
```
df_Test_Train.head(100)
```

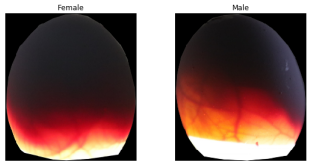
Out[4]:

	Crp_Filepath	clutch	Day	egg_number	sex
0	Cropped_Egg_images/Clutch2_D14/IMG_0002.JPG	2	14	1	Female
1	Cropped_Egg_images/Clutch2_D14/IMG_0003.JPG	2	14	2	Male
2	Cropped_Egg_images/Clutch2_D14/IMG_0007.JPG	2	14	6	Female
3	Cropped_Egg_images/Clutch2_D14/IMG_0008.JPG	2	14	7	Female
4	Cropped_Egg_images/Clutch2_D14/IMG_0010.JPG	2	14	9	Male
5	Cropped_Egg_images/Clutch2_D14/IMG_0013.JPG	2	14	12	Male
6	Cropped_Egg_images/Clutch2_D14/IMG_0014.JPG	2	14	13	Female
7	Cropped_Egg_images/Clutch2_D14/IMG_0016.JPG	2	14	15	Female
8	Cropped_Egg_images/Clutch2_D14/IMG_0018.JPG	2	14	17	Female
9	Cropped_Egg_images/Clutch2_D14/IMG_0022.JPG	2	14	21	Female
10	Cropped_Egg_images/Clutch2_D14/IMG_0025.JPG	2	14	24	Female

Initial testing

In [165]:
```
np.random.seed(42)
data = ImageDataBunch.from_df('/home/jplineb/Chicken_Proj', df_Test_Train, label_col = 'sex', size = (252,224), bs = 2, valid_pct=0.20, ds_tfms = None)
```

In [166]:
```
data.show_batch()
```



In [164]:
```
data.batch_stats
```

Out[164]:
```
<bound method ImageDataBunch.batch_stats of ImageDataBunch;

Train: LabelList (76 items)
x: ImageList
Image (3, 224, 252),Image (3, 224, 252),Image (3, 224, 252),Image (3, 224, 252),Image (3, 224, 252)
y: CategoryList
Female,Male,Female,Female,Female
Path: /home/jplineb/Chicken_Proj;

Valid: LabelList (18 items)
x: ImageList
Image (3, 224, 252),Image (3, 224, 252),Image (3, 224, 252),Image (3, 224, 252),Image (3, 224, 252)
y: CategoryList
Female,Female,Female,Female,Female
Path: /home/jplineb/Chicken_Proj;

Test: None>
```

In [174]:
```
# Attempting to use CSVLogger callback
learn = cnn_learner(data, models.resnet34, metrics = error_rate, wd=10, callback_fns=[CSVLogger])
```

In [175]:
```
learn.fit_one_cycle(6)
learn.recorder.plot_losses()
```

Total time: 00:16

epoch	train_loss	valid_loss	error_rate	time
0	0.868182	1.010944	0.611111	00:02
1	0.771957	0.717360	0.611111	00:02
2	0.722283	0.702474	0.722222	00:02
3	0.704669	0.696288	0.666667	00:02
4	0.697446	0.695262	0.666667	00:02
5	0.694409	0.695622	0.666667	00:02

In [176]:
```
# CSVLogger callback testing
df_history = pd.read_csv('history.csv')
df_history.head()
```

Out[176]:

	epoch	train_loss	valid_loss	error_rate	time
0	0	0.868182	1.010944	0.611111	NaN
1	1	0.771957	0.717360	0.611111	NaN
2	2	0.722283	0.702474	0.722222	NaN
3	3	0.704669	0.696288	0.666667	NaN
4	4	0.697446	0.695262	0.666667	NaN

In [178]:
```
df_history.error_rate.max()
```

Out[178]:
```
0.722222
```

In [191]:
```
## Testing dataframe creation ##
DFBig = pd.DataFrame(columns = ['model_arch', 'transforms', 'normalized', 'weight_decay', 'max_error', 'min_error', 'avg_error', 'train_df'])
DFBig = DFBig.append({'model_arch': 'resnet34',
              'transforms': 'None',
              'normalized': 'False',
              'weight_decay': int(10),
              'max_error' : df_history.error_rate.max(),
              'min_error': df_history.error_rate.min(),
              'avg_error': df_history.error_rate.mean(),
              'train_df': df_history}, ignore_index = True)
```

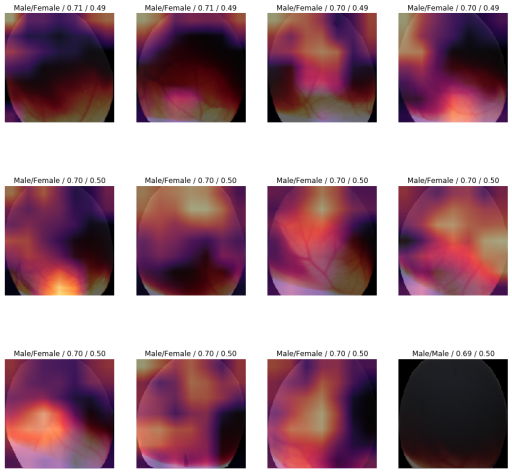
Out[191]:

	epoch	train_loss	valid_loss	error_rate	time
0	0	0.868182	1.010944	0.611111	NaN
1	1	0.771957	0.717360	0.611111	NaN
2	2	0.722283	0.702474	0.722222	NaN
3	3	0.704669	0.696288	0.666667	NaN
4	4	0.697446	0.695262	0.666667	NaN
5	5	0.694409	0.695622	0.666667	NaN

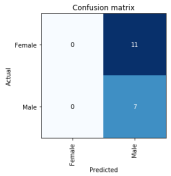
In [115]:
```
interp = ClassificationInterpretation.from_learner(learn)

interp.plot_top_losses(12, figsize=(15,15))
```

prediction/actual/loss/probability



In [196]:
```
alpha = interp.plot_confusion_matrix(return_fig=True).savefig('test.png')
```



For Loop

In [ ]:
```
### A Function to test multiple parameters for training fastai models  ## Second Technique
'''
Things we want to vary:
    Model Architecture,Transforms,Normalization,Weight Decay
Things we want to store:
    Training Results, lowest error, individual model(unfrozen), confusion matrix
'''

# Parameters to very
modelarch = [models.resnet18, models.resnet34, models.resnet50]
transforms = [None, get_transforms()]
normalization = [True, False]
weight_decay = [.01, .1, 1, 10, 20]
epoch_cycles = 7

# Creating Frame Work
DFBig = pd.DataFrame(columns = ['model_arch', 'transforms', 'normalized', 'weight_decay', 'max_error', 'min_error', 'avg_error', 'train_df'])
np.random.seed(42)
for trans in transforms:
    for norm in normalization:
        if norm:
            data = ImageDataBunch.from_df('/home/jplineb/Chicken_Proj', df_Test_Train, label_col = 'sex', size = 224, bs = 2, valid_pct=0.20, ds_tfms = trans).normalize()
        else:
            data = ImageDataBunch.from_df('/home/jplineb/Chicken_Proj', df_Test_Train, label_col = 'sex', size = 224, bs = 2, valid_pct=0.20, ds_tfms = trans)
        for arch in modelarch:
            archstr = str(arch).split(' ')[1]
            for wd in weight_decay:
                if trans is not None:
                    trans = 'stock'
                test_name =  archstr + '_' + str(trans) + '_' + str(norm) + '_' + str(wd)
                learn = cnn_learner(data, arch, metrics = error_rate, wd=wd, callback_fns=[CSVLogger])
                learn.fit_one_cycle(epoch_cycles)
                learn.save(test_name)
                df_history = pd.read_csv('history.csv')
                DFBig = DFBig.append({'model_arch': archstr,
                                      'transforms': str(trans),
                                      'normalized': str(norm),
                                      'weight_decay': wd,
                                      'max_error' : df_history.error_rate.max(),
                                      'min_error': df_history.error_rate.min(),
                                      'avg_error': df_history.error_rate.mean(),
                                      'train_df': df_history}, ignore_index = True)
                interp = ClassificationInterpretation.from_learner(learn)
                interp.plot_confusion_matrix(return_fig=True)#.savefig('./matrices/' + test_name + '.png'))
                DFBig.to_csv('DF_Big.csv')
```

Results from this test found Resnet50 and Resnet34 needed further testing

df_agg_clean - DataFrame

Index	('min_error', 'average')	('min_error', '<lambda>')	('max_mean_error', '')
resnet50_0.01_modified_True	0.308772	0.0379857	0.384743
resnet50_1_modified_True	0.329825	0.0273594	0.384543
resnet34_0.003_modified_False	0.340351	0.0235899	0.387531
resnet34_0.01_None_False	0.340935	0.0140741	0.369084
resnet50_0.01_Stock_True	0.360819	0.0295145	0.419848
resnet34_0.1_modified_False	0.361403	0.0265371	0.414477
resnet18_5_None_False	0.361403	0.0450984	0.4516
resnet34_0.003_None_True	0.361403	0.0219686	0.405341
resnet50_5_None_True	0.361988	0.0483254	0.458639
resnet18_0.003_Stock_True	0.362573	0.0365017	0.435576
resnet18_0.01_Stock_False	0.362573	0.0515948	0.465763
resnet18_0.1_Stock_False	0.362573	0.025837	0.414247
resnet34_5_None_True	0.362573	0.0394205	0.441414
resnet34_0.01_modified_True	0.362573	0.0365017	0.435577
resnet50_0.003_Stock_True	0.37076	0.0523214	0.475403
resnet34_1_modified_False	0.371345	0.0378811	0.447107
resnet18_1_Stock_False	0.374269	0.0415494	0.457368
resnet50_0.01_None_True	0.382456	0.0334348	0.449325
resnet50_1_Stock_True	0.383041	0.0544583	0.491957
resnet34_0.1_None_False	0.383041	0.017505	0.418051

Format | Resize | Background color | Column min/max | Save and Close | Close

For Further testing of these parameters look at: Repeated Stratified K Fold on "Good" hyperparametrs (Repeated%20Stratified%20KFold.ipynb)

In [15]:
```
## Test of Cross Validation Loading
# This was used for learning how to use sklearn's stratified KFold
from sklearn.model_selection import KFold
tfms = get_transforms()
kf = KFold(n_splits=2)
acc_val = []
kappa = KappaScore()
kappa.weights = "quadratic"
for train_index, val_index in kf.split(df_Test_Train.index):
    print(train_index, val_index)
    data_fold = (ImageList.from_df(df_Test_Train,'/home/jplineb/Chicken_Proj')
                .split_by_idxs(train_index, val_index)
                .label_from_df(cols='sex')
                .transform(tfms, size=224)
                .databunch(bs = 2)).normalize()
    learn = cnn_learner(data_fold, models.resnet18, metrics=error_rate, wd=.1)
    learn.fit_one_cycle(7)
    loss, acc = learn.validate()
    acc_val.append(acc.numpy())
```

In [7]:
```
from sklearn.metrics import roc_auc_score

def auroc_score(input, target):
    input, target = input.cpu().numpy()[:,1], target.cpu().numpy()
    return roc_auc_score(target, input)

class AUROC(Callback):
    _order = -20 #Needs to run before the recorder

    def __init__(self, learn,extra=None, **kwargs): self.learn = learn
    def on_train_begin(self, **kwargs): self.learn.recorder.add_metric_names(['AUROC'])
    def on_epoch_begin(self, **kwargs): self.output, self.target = [], []

    def on_batch_end(self, last_target, last_output, train, **kwargs):
        if not train:
            self.output.append(last_output)
            self.target.append(last_target)

    def on_epoch_end(self, last_metrics, **kwargs):
        if len(self.output) > 0:
            output = torch.cat(self.output)
            target = torch.cat(self.target)
            preds = F.softmax(output, dim=1)
            metric = auroc_score(preds, target)
            return add_metrics(last_metrics, [metric])
```

In [ ]:
```
# Itterative method with Cross Validation
from sklearn.model_selection import StratifiedKFold

# Parameters to very
modelarch = [models.resnet18, models.resnet34, models.resnet50]
transforms = [None, get_transforms(),
              get_transforms(do_flip = True, flip_vert = False, max_rotate = 35, max_lighting = None, max_warp = .2, p_lighting = 0)]
normalization = [True, False]
weight_decay = [.01,.1,.3,.5,1]
epoch_cycles = 9

# Creating Frame Work
DFBig = pd.DataFrame(columns = ['test_name','model_arch', 'transforms', 'normalized',
                                'weight_decay', 'split_num'])
np.random.seed(42) #locks random seed
skf = StratifiedKFold(n_splits=5)
for arch in modelarch:
    archstr = str(arch).split(' ')[1]
    for wd in weight_decay:
        #kf = KFold(n_splits=5) # creates k fold crosvalidation with n splits
        #acc_val = [] # creates an empy list to store accuracy
        #test_name = archstr + '_' + 'tsf' + '_' + 'norm' + '_' + str(wd) # this run Locks in the transforms and normalization; that variability comes in in later tests

        for norm in normalization:

            for tsfms in transforms:
                if tsfms is not None:
                    if tsfms == get_transforms():
                        tsfmstr = 'Stock'
                    else:
                        tsfmstr = 'modified'
                else:
                    tsfmstr = 'None'

                test_name = archstr + '_' + str(wd) + '_' + tsfmstr + '_' + str(norm)
                print(test_name)
                split_num = 1
                #for train_index, val_index in kf.split(df_Test_Train.index):
                for train_index, val_index in skf.split(df_Test_Train.index, df_Test_Train['sex']):
                    if norm:
                        data_fold = (ImageList.from_df(df_Test_Train,'/home/jplineb/Chicken_Proj')
                                .split_by_idxs(train_index, val_index)
                                .label_from_df(cols='sex')
                                .transform(tsfms, size=224)
                                .databunch(bs = 2)).normalize()
                    else:
                        data_fold = (ImageList.from_df(df_Test_Train,'/home/jplineb/Chicken_Proj')
                                .split_by_idxs(train_index, val_index)
                                .label_from_df(cols='sex')
                                .transform(tsfms, size=224)
                                .databunch(bs = 2))

                    learn = cnn_learner(data_fold, arch, metrics=error_rate, callback_fns = [CSVLogger,AUROC, partial(EarlyStoppingCallback, monitor='AUROC', mode='max', min_delta=0.01, patience=100)], wd=wd)
                    learn.fit_one_cycle(epoch_cycles)

                    df_history = pd.read_csv('history.csv') # adds to dataframec created earlier (again norm and tsfms locked to ON)
                    DFBig = DFBig.append({'test_name': test_name,
                            'model_arch': archstr,
                            'transforms': tsfmstr,
                            'normalized': str(norm),
                            'weight_decay': wd,
                            'split_num': split_num, # indicates kfold split
                            'error_rate_8' : df_history.error_rate[7],
                            'error_rate_9' : df_history.error_rate[8],
                            'AUROC_8': df_history.AUROC[7],
                            'AUROC_9': df_history.AUROC[8],
                            'train_df': df_history}, ignore_index = True)

                    split_num+=1
                    #interp = ClassificationInterpretation.from_Learner(learn)
                    #interp.plot_confusion_matrix(return_fig=True, title =test_name)#.savefig(('./matrices/' + test_name + '.png'))
                    DFBig.to_csv('DF_Big.csv')
```

```
resnet18_0.01_None_True
```

Total time: 00:13

epoch	train_loss	valid_loss	error_rate	AUROC	time
0	1.091993	0.868008	0.750000	0.437500	00:01
1	0.938231	0.846525	0.625000	0.375000	00:01
2	0.937822	1.076271	0.625000	0.437500	00:01
3	0.834414	0.925072	0.250000	0.625000	00:01
4	0.836165	0.761775	0.375000	0.625000	00:01
5	0.752544	0.844406	0.500000	0.625000	00:01
6	0.708230	0.838999	0.500000	0.625000	00:01
7	0.707960	0.933295	0.375000	0.625000	00:01
8	0.751757	0.764272	0.500000	0.625000	00:01

Results from the testing

df_mean - DataFrame

Index	Unnamed: 0	weight_decay	split_num	AUROC_8	AUROC_9	error_rate_8	error_rate_9
resnet18_0.01_None_False	17	0.01	3	0.670833	0.620833	0.410714	0.382143
resnet50_0.3_None_True	362	0.3	3	0.654167	0.666667	0.360714	0.385714
resnet50_0.5_modified_T…	399.5	0.5	3	0.652083	0.583333	0.451786	0.435714
resnet50_0.01_None_False	317	0.01	3	0.633333	0.6875	0.407143	0.407143
resnet50_1_None_False	437	1	3	0.608333	0.6	0.442857	0.439286
resnet50_0.3_None_False	377	0.3	3	0.591667	0.570833	0.564286	0.539286
resnet50_0.01_modified_…	324.5	0.01	3	0.5875	0.597917	0.430357	0.455357
resnet18_1_modified_Fal…	144.5	1	3	0.5875	0.591667	0.425	0.446429
resnet50_0.5_modified_F…	414.5	0.5	3	0.575	0.591667	0.45	0.5
resnet18_0.01_Stock_True	7	0.01	3	0.566667	0.5375	0.485714	0.435714
resnet34_0.1_None_False	197	0.1	3	0.558333	0.558333	0.457143	0.432143
resnet18_0.3_modified_F…	84.5	0.3	3	0.55625	0.55625	0.476786	0.4625
resnet50_0.1_None_False	347	0.1	3	0.55	0.608333	0.485714	0.485714
resnet50_0.5_None_True	392	0.5	3	0.541667	0.525	0.439286	0.489286
resnet18_0.5_modified_T…	99.5	0.5	3	0.5375	0.539583	0.491071	0.491071
resnet18_1_None_True	122	1	3	0.533333	0.545833	0.460714	0.464286
resnet34_0.01_None_True	152	0.01	3	0.533333	0.5375	0.592857	0.485714
resnet34_0.01_None_False	167	0.01	3	0.529167	0.583333	0.535714	0.535714
resnet50_0.5_None_False	407	0.5	3	0.529167	0.529167	0.407143	0.435714
resnet34_0.1_modified_T…	189.5	0.1	3	0.525	0.491667	0.553571	0.501786

Format | Resize | Background color | Column min/max | Save and Close | Close

Best Results

- Resnet18
- No Transforms
- wd = 0.01
- 9 epochs
- normalization off
- max lr not set

Search for Repeated Stratified K-Fold on best results (Previous%20best%20and%20new%20hyperparameter%20search.ipynb)

Devils Advocate Testing

Hypothesis: Flip day 1 gender

In [137]:
```
def changegend(row):
    index = row.name
    gend = row['sex']
    if index < 55:
        if gend=='Female':
            newgend = 'Male'
        if gend=='Male':
            newgend = 'Female'
    else:
        newgend = gend
    return (newgend)

df_swapped = pd.DataFrame(df_Test_Train)
df_swapped['sex'] = df_swapped.apply(changegend, axis = 1)
df_swapped.head(100)
```

Out[137]:

	Crp_Filepath	clutch	Day	egg_number	sex
0	Cropped_Egg_images/Clutch1_D18/egg2.JPG	1	18	2	Male
1	Cropped_Egg_images/Clutch1_D18/egg3.JPG	1	18	3	Male
2	Cropped_Egg_images/Clutch1_D18/egg4.JPG	1	18	4	Female
3	Cropped_Egg_images/Clutch1_D18/egg6.JPG	1	18	6	Male
4	Cropped_Egg_images/Clutch1_D18/egg9.JPG	1	18	9	Male
5	Cropped_Egg_images/Clutch1_D18/egg11.JPG	1	18	11	Male
6	Cropped_Egg_images/Clutch1_D18/egg15.JPG	1	18	15	Male
7	Cropped_Egg_images/Clutch1_D18/egg16.JPG	1	18	16	Female
8	Cropped_Egg_images/Clutch1_D18/egg17.JPG	1	18	17	Male
9	Cropped_Egg_images/Clutch1_D18/egg19.JPG	1	18	19	Female
10	Cropped_Egg_images/Clutch1_D18/egg20.JPG	1	18	20	Female

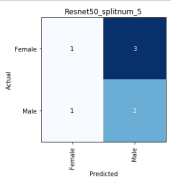
##

In [154]:
```
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

import numpy as np
from sklearn.model_selection import StratifiedKFold
np.random.seed(42)
tfms = get_transforms(do_flip = True, flip_vert = False, max_rotate = 35, max_lighting = None, max_warp = .2, p_lighting = 0)
kf = KFold(n_splits=5)
#skf = StratifiedKFold(n_splits=5, shuffle = False)

acc_val = []
split_num = 1
## Attempt to use Just Clutch2 Data
df_Test_Train_Clutch2 = df_Test_Train[55:]
#df_Test_Train_Clutch1 = df_Test_Train[:55]
#for train_index, val_index in skf.split(df_Test_Train.index, df_Test_Train['sex']):
#for train_index, val_index in kf.split(df_Test_Train.index):
#for train_index, val_index in kf.split(df_Test_Train_Clutch2.index):
for train_index, val_index in kf.split(df_swapped.index):
    print(train_index, val_index)
    test_name = ("Resnet50_splitnum_" + str(split_num))
    data_fold = (ImageList.from_df(df_Test_Train,'/home/jplineb/Chicken_Proj')
                .split_by_idxs(train_index, val_index)
                .label_from_df(cols='sex')
                .transform(tfms, size=224)
                .databunch(bs = 2)).normalize()
    learn = cnn_learner(data_fold, models.resnet18, metrics=error_rate, wd=.01)
    learn.save(test_name)
    learn.fit_one_cycle(7)
    loss, acc = learn.validate()
    learn.show_results()
    interp = ClassificationInterpretation.from_learner(learn)
    interp.plot_confusion_matrix(return_fig=True, title =test_name)
    acc_val.append(acc.numpy())
    split_num +=1
```

Out[154]:



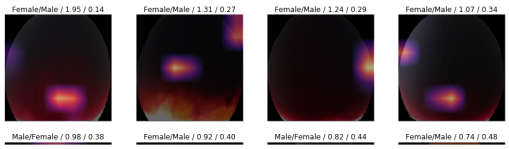
In [152]:
```
print('Average error rate:' + str(1-np.average(acc_val)))
print('Standard error bounds:' + str(np.std(acc_val)/np.sqrt(5)))
```

```
Average error rate:0.44502925872802734
Standard error bounds:0.04633114316537905
```

In [61]:
```
interp = ClassificationInterpretation.from_learner(learn.load('Resnet50_splitnum_5'))

interp.plot_top_losses(16, figsize=(15,15))
```

prediction/actual/loss/probability



K-Fold Stratified Shuffle