Vista

August 26, 2019

[1]:
```
# This Python 3 environment comes with many helpful analytics libraries␣
 ↪installed
# It is defined by the kaggle/python docker image: https://github.com/kaggle/
 ↪docker-python
# For example, here's several helpful packages to load in

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) will list␣
 ↪all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# Any results you write to the current directory are saved as output.
```

```
/kaggle/input/vistacodefest19-cellular-image-classification/dc99623a-c-
vista_data/test_final.zip
/kaggle/input/vistacodefest19-cellular-image-classification/dc99623a-c-
vista_data/sample.csv
/kaggle/input/vistacodefest19-cellular-image-classification/dc99623a-c-
vista_data/train_final.zip
/kaggle/input/vistacodefest19-cellular-image-classification/dc99623a-c-
vista_data/train.csv
```

[2]:
```
!unzip /kaggle/input/vistacodefest19-cellular-image-classification/
 ↪dc99623a-c-vista_data/train_final.zip -d /kaggle/train_final
```

```
Archive:  /kaggle/input/vistacodefest19-cellular-image-
classification/dc99623a-c-vista_data/train_final.zip
  inflating: /kaggle/train_final/f1.png
  inflating: /kaggle/train_final/f10.png
  inflating: /kaggle/train_final/f100.png
  inflating: /kaggle/train_final/f1000.png
```

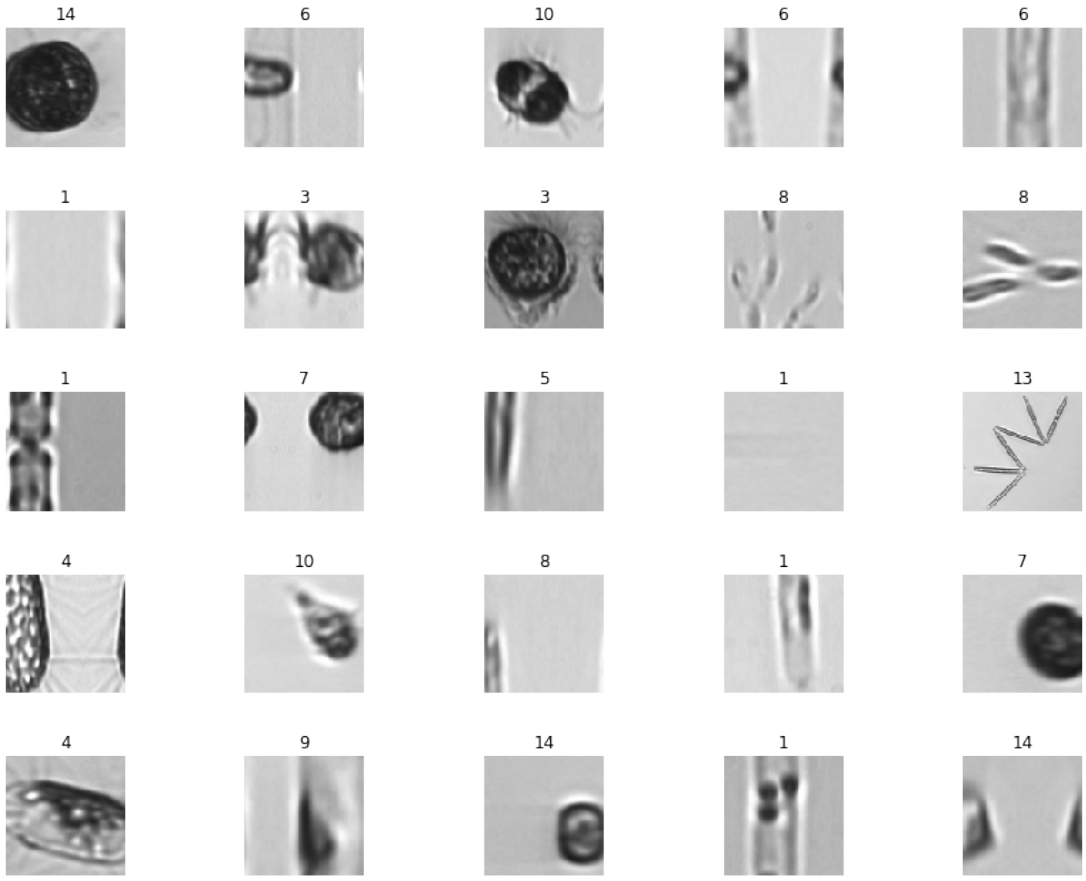
```
x: ImageList
Image (3, 96, 96),Image (3, 96, 96),Image (3, 96, 96),Image (3, 96, 96),Image
(3, 96, 96)
y: CategoryList
4,5,2,14,4
Path: /kaggle/train_final;

Test: None
```

[14]:
```
from torchvision.models import *
arch = densenet121
acc_02 = partial(accuracy_thresh, thresh=0.2)
acc_03 = partial(accuracy_thresh, thresh=0.3)
acc_04 = partial(accuracy_thresh, thresh=0.4)
acc_05 = partial(accuracy_thresh, thresh=0.5)
f_score = partial(fbeta, thresh=0.2)
learn = cnn_learner(data, arch, metrics=[accuracy, FBeta('macro')])
```

```
Downloading: "https://download.pytorch.org/models/densenet121-a639ec97.pth" to
/tmp/.cache/torch/checkpoints/densenet121-a639ec97.pth
100%|      | 30.8M/30.8M [00:00<00:00, 120MB/s]
```

[17]:
```
ts_path = Path('/kaggle/test_final/')
test_imgs = ts_path.ls()
test_imgs.sort(key=lambda x: x.stem)
data.add_test(test_imgs)
learn.data = data
preds = learn.get_preds(ds_type=DatasetType.Test)
```

[18]:
```
data.show_batch(rows=5, figsize=(12,9))
```

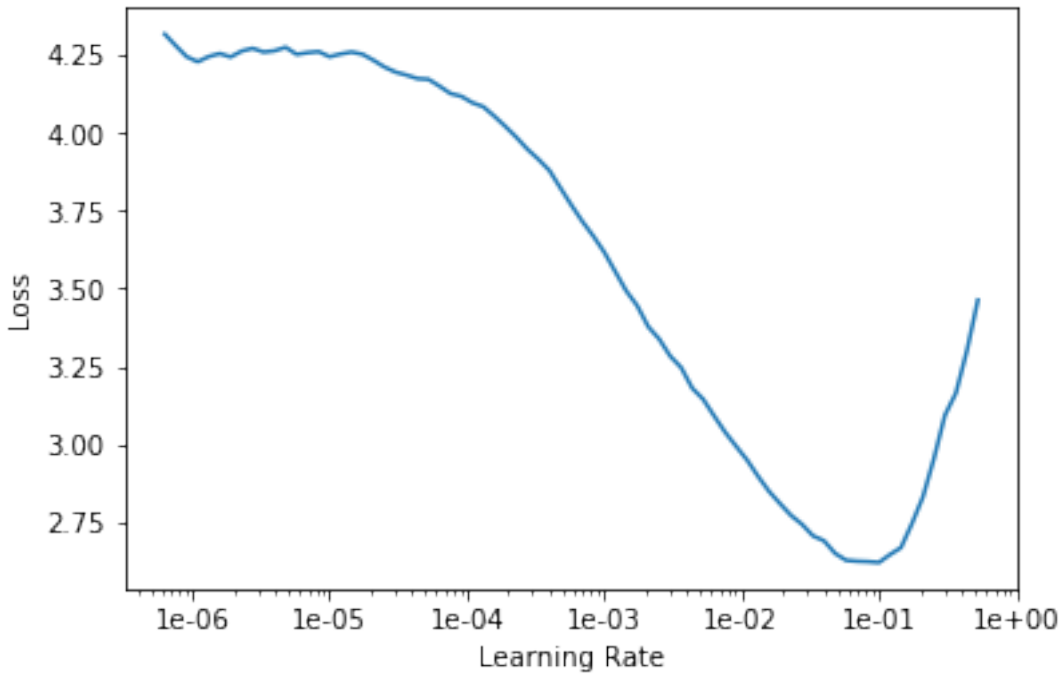


[19]:
```
learn.lr_find()
```

```
<IPython.core.display.HTML object>
```

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

[20]:
```
learn.recorder.plot()
```



[21]:
```
lr = 5e-2
```

[22]:
```
learn.fit_one_cycle(4, slice(lr))
```

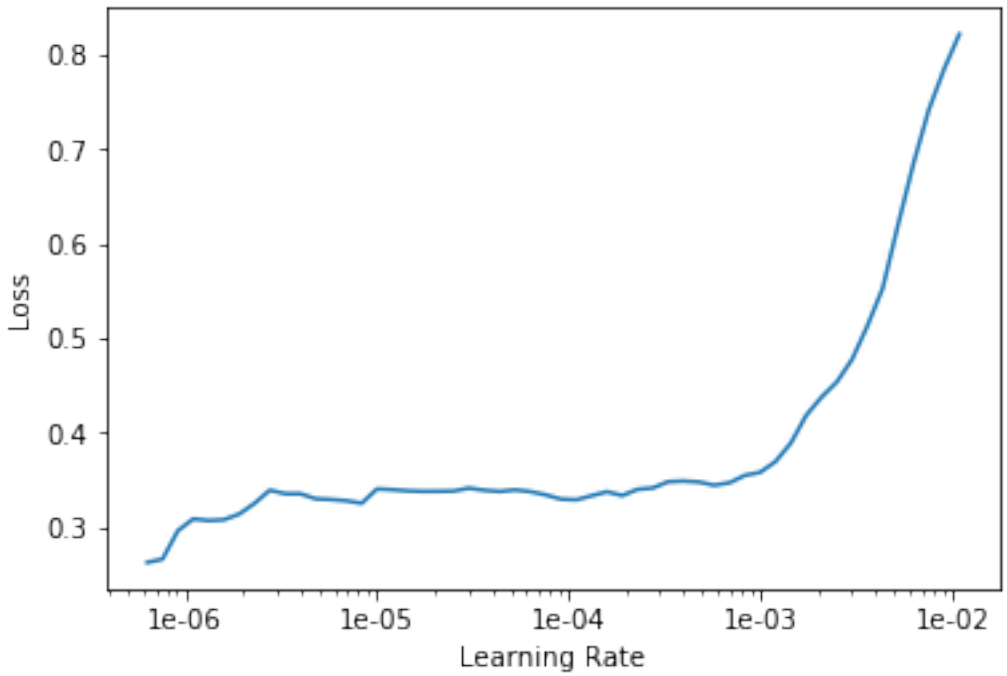
```
<IPython.core.display.HTML object>
```

[23]:
```
learn.save('stage1')
```

[24]:
```
learn.unfreeze()
learn.lr_find()
learn.recorder.plot()
```

```
<IPython.core.display.HTML object>
```

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

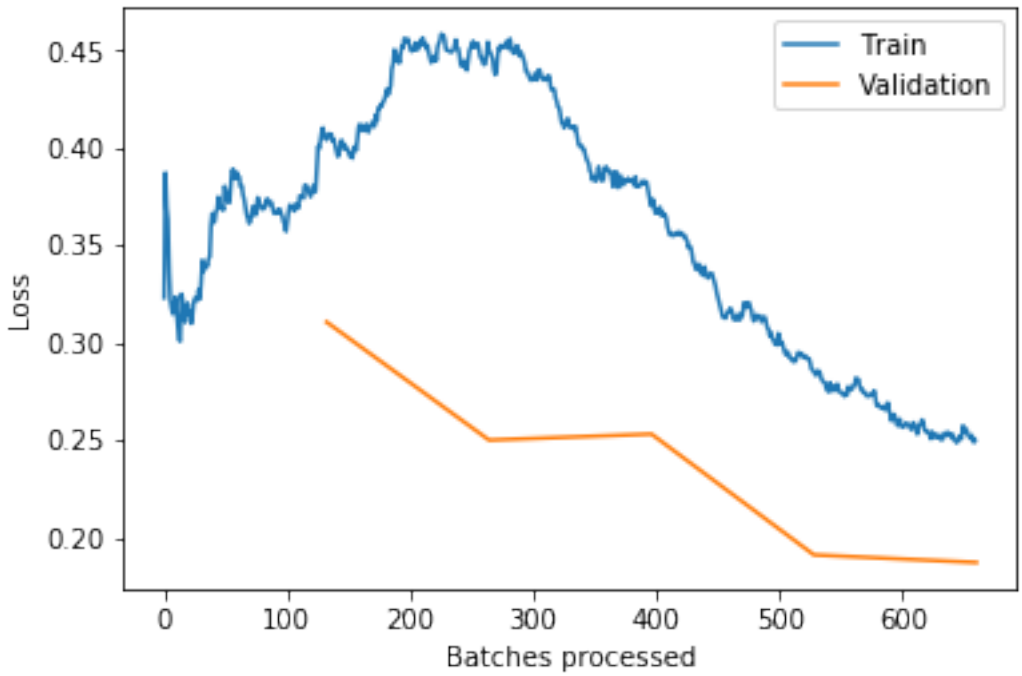


[25]:
```
learn.fit_one_cycle(5, slice(1e-5, lr/5))
```

```
<IPython.core.display.HTML object>
```

[26]:
```
learn.save('stage2')
```

[27]:
```
learn.recorder.plot_losses()
```



[28]:
```
sub=pd.read_csv('/kaggle/input/vistacodefest19-cellular-image-classification/
↪dc99623a-c-vista_data/sample.csv').set_index('id')
sub.head()
```

[28]:
```
         label
id
f1.png       1
f2.png       1
f3.png       1
f4.png       1
f5.png       1
```

[29]:
```
pred_test,y_test = learn.get_preds(DatasetType.Test)
pred_score = accuracy(pred_test,y_test)
```

[30]:
```
pred_test_tta,y_test_tta = learn.TTA(ds_type=DatasetType.Test)
pred_score_tta = accuracy(pred_test_tta,y_test_tta)
```

```
<IPython.core.display.HTML object>
```

[31]:
```
pred_test_tta.shape
```

[31]:
```
torch.Size([15984, 14])
```

[32]:
```
y_test_tta.shape
```

[32]:
```
torch.Size([15984])
```

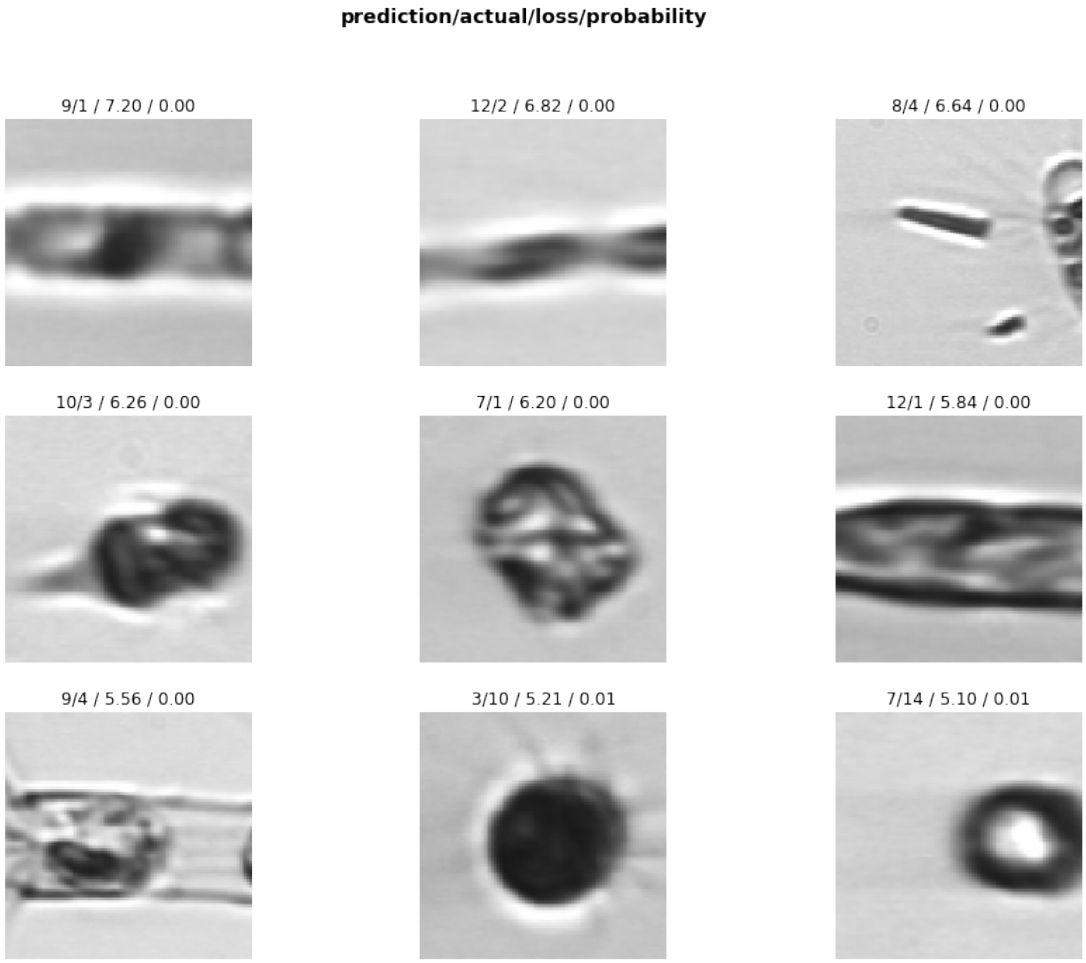
[33]:
```
clean_fname=np.vectorize(lambda fname: str(fname).split('/')[-1])
fname_cleaned=clean_fname(data.test_ds.items)
fname_cleaned=fname_cleaned.astype(str)
```

[34]:
```
data
```

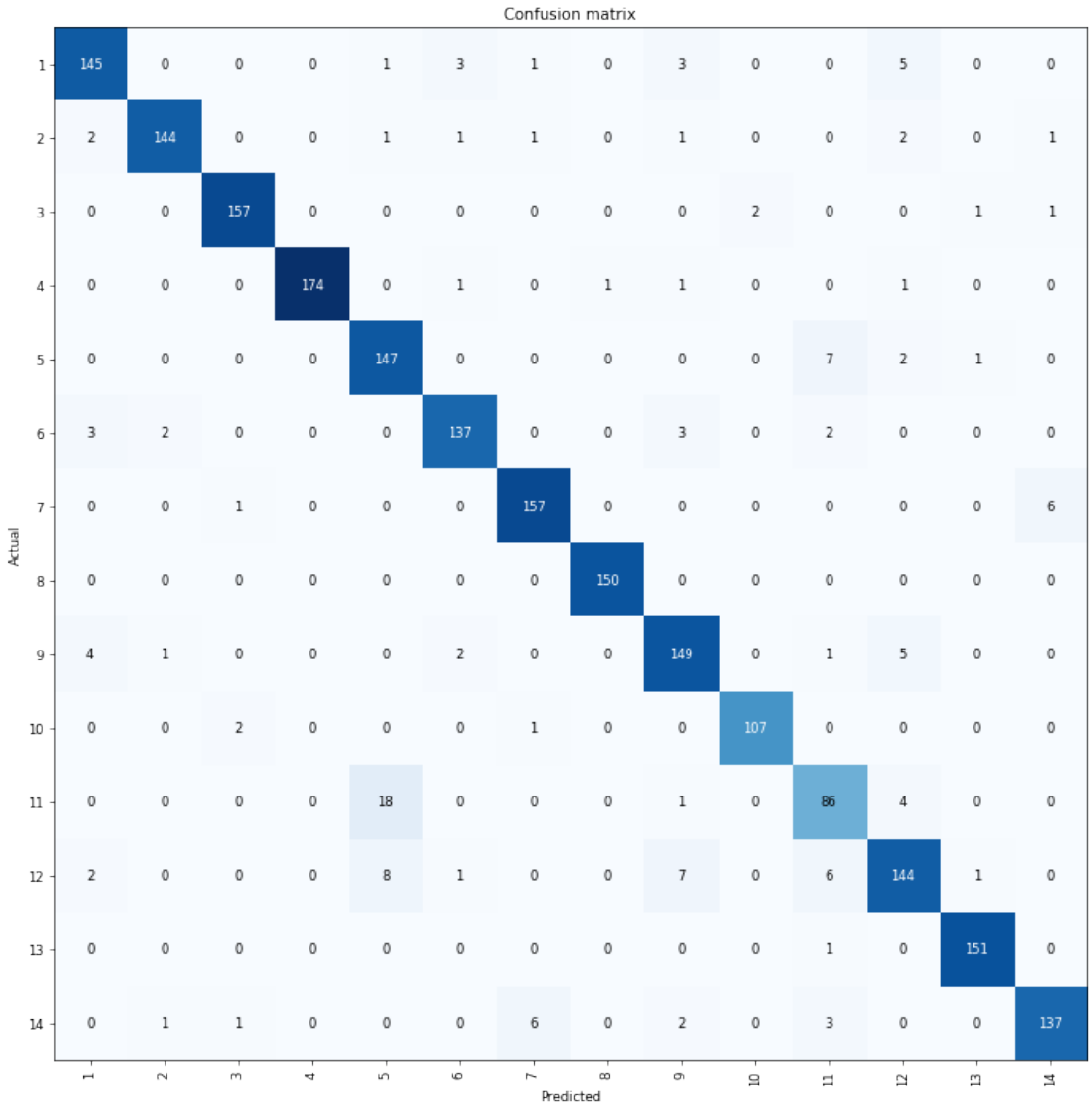
[34]:
```
ImageDataBunch;

Train: LabelList (8480 items)
x: ImageList
Image (3, 96, 96),Image (3, 96, 96),Image (3, 96, 96),Image (3, 96, 96),Image
(3, 96, 96)
y: CategoryList
10,7,7,7,10
Path: /kaggle/train_final;

Valid: LabelList (2120 items)
x: ImageList
Image (3, 96, 96),Image (3, 96, 96),Image (3, 96, 96),Image (3, 96, 96),Image
(3, 96, 96)
y: CategoryList
4,5,2,14,4
Path: /kaggle/train_final;

Test: LabelList (15984 items)
x: ImageList
Image (3, 96, 96),Image (3, 96, 96),Image (3, 96, 96),Image (3, 96, 96),Image
(3, 96, 96)
y: EmptyLabelList
,,,,
Path: /kaggle/train_final
```

[35]:
```
interp = ClassificationInterpretation.from_learner(learn)
```

[36]:
```
interp.plot_top_losses(9, figsize=(15,11))
```



[37]:
```
interp.plot_confusion_matrix(figsize=(12,12), dpi=60)
```



[38]:
```
interp.most_confused(min_val=2)
```

[38]:
```
[(11, 5, 18),
 (12, 5, 8),
 (5, 11, 7),
 (12, 9, 7),
 (7, 14, 6),
 (12, 11, 6),
 (14, 7, 6),
 (1, 12, 5),
 (9, 12, 5),
 (9, 1, 4),
```

```
 (11, 12, 4),
 (1, 6, 3),
 (1, 9, 3),
 (6, 1, 3),
 (6, 9, 3),
 (14, 11, 3),
 (2, 1, 2),
 (2, 12, 2),
 (3, 10, 2),
 (5, 12, 2),
 (6, 2, 2),
 (6, 11, 2),
 (9, 6, 2),
 (10, 3, 2),
 (12, 1, 2),
 (14, 9, 2)]
```

[39]:
```
thresh = 0.2
labelled_preds = [' '.join([str(learn.data.classes[i]) for i,p in␣
 ↪enumerate(pred) if p > thresh]) for pred in pred_test]
fnames = [f.name[:-4] for f in learn.data.test_ds.items]
```

[40]:
```
sub.loc[fname_cleaned,'label']=to_np(pred_test[:,1])
sub.to_csv(f'submission_{pred_score}.csv')
sub.loc[fname_cleaned,'label']=to_np(pred_test_tta[:,1])
sub.to_csv(f'submission_{pred_score_tta}.csv')
```

```
/opt/conda/lib/python3.6/site-
packages/tensorflow/python/framework/dtypes.py:516: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/opt/conda/lib/python3.6/site-
packages/tensorflow/python/framework/dtypes.py:517: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/opt/conda/lib/python3.6/site-
packages/tensorflow/python/framework/dtypes.py:518: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
/opt/conda/lib/python3.6/site-
packages/tensorflow/python/framework/dtypes.py:519: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
```

```
/opt/conda/lib/python3.6/site-
packages/tensorflow/python/framework/dtypes.py:520: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
/opt/conda/lib/python3.6/site-
packages/tensorflow/python/framework/dtypes.py:525: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
/opt/conda/lib/python3.6/site-
packages/tensorboard/compat/tensorflow_stub/dtypes.py:541: FutureWarning:
Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/opt/conda/lib/python3.6/site-
packages/tensorboard/compat/tensorflow_stub/dtypes.py:542: FutureWarning:
Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/opt/conda/lib/python3.6/site-
packages/tensorboard/compat/tensorflow_stub/dtypes.py:543: FutureWarning:
Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
/opt/conda/lib/python3.6/site-
packages/tensorboard/compat/tensorflow_stub/dtypes.py:544: FutureWarning:
Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
/opt/conda/lib/python3.6/site-
packages/tensorboard/compat/tensorflow_stub/dtypes.py:545: FutureWarning:
Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
/opt/conda/lib/python3.6/site-
packages/tensorboard/compat/tensorflow_stub/dtypes.py:550: FutureWarning:
Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
```

[41]:
```
len(pred_test[:,-1])
```

[41]: 15984

[42]:
```
!ls
```

```
__notebook_source__.ipynb          submission_0.07920420169830322.csv
```

```
submission_0.07745245099067688.csv
```

[43]:
```
# import the modules we'll need
from IPython.display import HTML
import pandas as pd
import numpy as np
import base64

# function that takes in a dataframe and creates a text link to
# download it (will only work for files < 2MB or so)
def create_download_link(df, title = "Download CSV file", filename = "data.
 ↪csv"):
    csv = df.to_csv()
    b64 = base64.b64encode(csv.encode())
    payload = b64.decode()
    html = '<a download="{filename}" href="data:text/csv;base64,{payload}"␣
 ↪target="_blank">{title}</a>'
    html = html.format(payload=payload,title=title,filename=filename)
    return HTML(html)

# create a random sample dataframe
df = pd.DataFrame(np.random.randn(50, 4), columns=list('ABCD'))

# create a link to download the dataframe
create_download_link(sub)

# ↓ ↓ ↓  Yay, download link! ↓ ↓ ↓
```

[43]:
```
<IPython.core.display.HTML object>
```

[ ]:

[ ]: