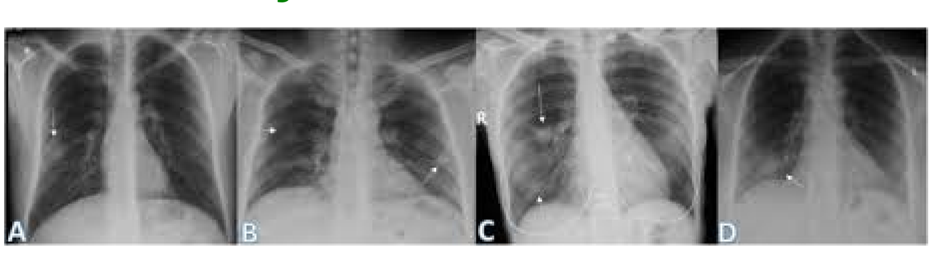
Chest X-ray Abnormalities Detection



[Author: Jitshil]

About competition

When you have a broken arm, radiologists help save the day—and the bone. These doctors diagnose and treat medical conditions using imaging techniques like CT and PET scans, MRIs, and, of course, X-rays. Yet, as it happens when working with such a wide variety of medical tools, radiologists face many daily challenges, perhaps the most difficult being the chest radiograph. The interpretation of chest X-rays can lead to medical misdiagnosis, even for the best practicing doctor. Computer-aided detection and diagnosis systems (CADe/CADx) would help reduce the pressure on doctors at metropolitan hospitals and improve diagnostic quality in rural areas.

Existing methods of interpreting chest X-ray images classify them into a list of findings. ***There is currently no specification of their locations on the image which sometimes leads to inexplicable results***. A solution for localizing findings on chest X-ray images is needed for providing doctors with more meaningful diagnostic assistance.

In this competition:

- **Task**: Automatically localize and classify 14 types of thoracic abnormalities from chest radiographs.

|Class-id| - |Class Name|

- 0 - Aortic enlargement
- 1 - Atelectasis
- 2 - Calcification
- 3 - Cardiomegaly
- 4 - Consolidation
- 5 - ILD
- 6 - Infiltration
- 7 - Lung Opacity
- 8 - Nodule/Mass
- 9 - Other lesion
- 10 - Pleural effusion
- 11 - Pleural thickening
- 12 - Pneumothorax
- 13 - Pulmonary fibrosis
- 14 - No Finding

Work Flow

Import Libs.

In [1]:
```
import os
import time
import random
import pydicom
import cv2
import numpy as np
import pandas as pd
from glob import glob
import matplotlib.pyplot as plt
from random import randint
from pydicom.pixel_data_handlers.util import apply_voi_lut
```

Woring Directory

In [2]:
```
train_dir = "../input/vinbigdata-chest-xray-abnormalities-detection/train"
test_dir = "../input/vinbigdata-chest-xray-abnormalities-detection/test"

train_files = os.listdir(train_dir)
test_files = os.listdir(test_dir)

train_df = pd.read_csv("../input/vinbigdata-chest-xray-abnormalities-detection/train.csv")
```

In [3]:
```
def get_bbox_area(row):
    return (row['x_max']-row['x_min'])*(row['y_max']-row['y_min'])


new_df = train_df[train_df['class_name'] != 'No finding']
new_df['bbox_area'] = new_df.apply(get_bbox_area, axis=1)
new_df.head()
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexin
g.html#returning-a-view-versus-a-copy
```

Out[3]:

	image_id	class_name	class_id	rad_id	x_min	y_min	x_max	y_max	bbox_area
2	9a5094b2563a1ef3ff50dc5c7ff71345	Cardiomegaly	3	R10	691.0	1375.0	1653.0	1831.0	438672.0
3	051132a778e61a86eb147c7c6f564dfe	Aortic enlargement	0	R10	1264.0	743.0	1611.0	1019.0	95772.0
5	1c32170b4af4ce1a3030eb8167753b06	Pleural thickening	11	R9	627.0	357.0	947.0	433.0	24320.0
6	0c7a38f293d5f5e4846aa4ca6db4daf1	ILD	5	R17	1347.0	245.0	2188.0	2169.0	1618084.0
7	47ed17dcb2cbeec15182ed335a8b5a9e	Nodule/Mass	8	R9	557.0	2352.0	675.0	2484.0	15576.0

DICOM image

Digital Imaging and Communications in Medicine (DICOM) is the standard for the communication and management of medical imaging information and related data. DICOM is most commonly used for storing and transmitting medical images enabling the integration of medical imaging devices such as scanners, servers, workstations, printers, network hardware, and picture archiving and communication systems (PACS) from multiple manufacturers. It has been widely adopted by hospitals and is making inroads into smaller applications like dentists' and doctors' offices.

DICOM to numpy array

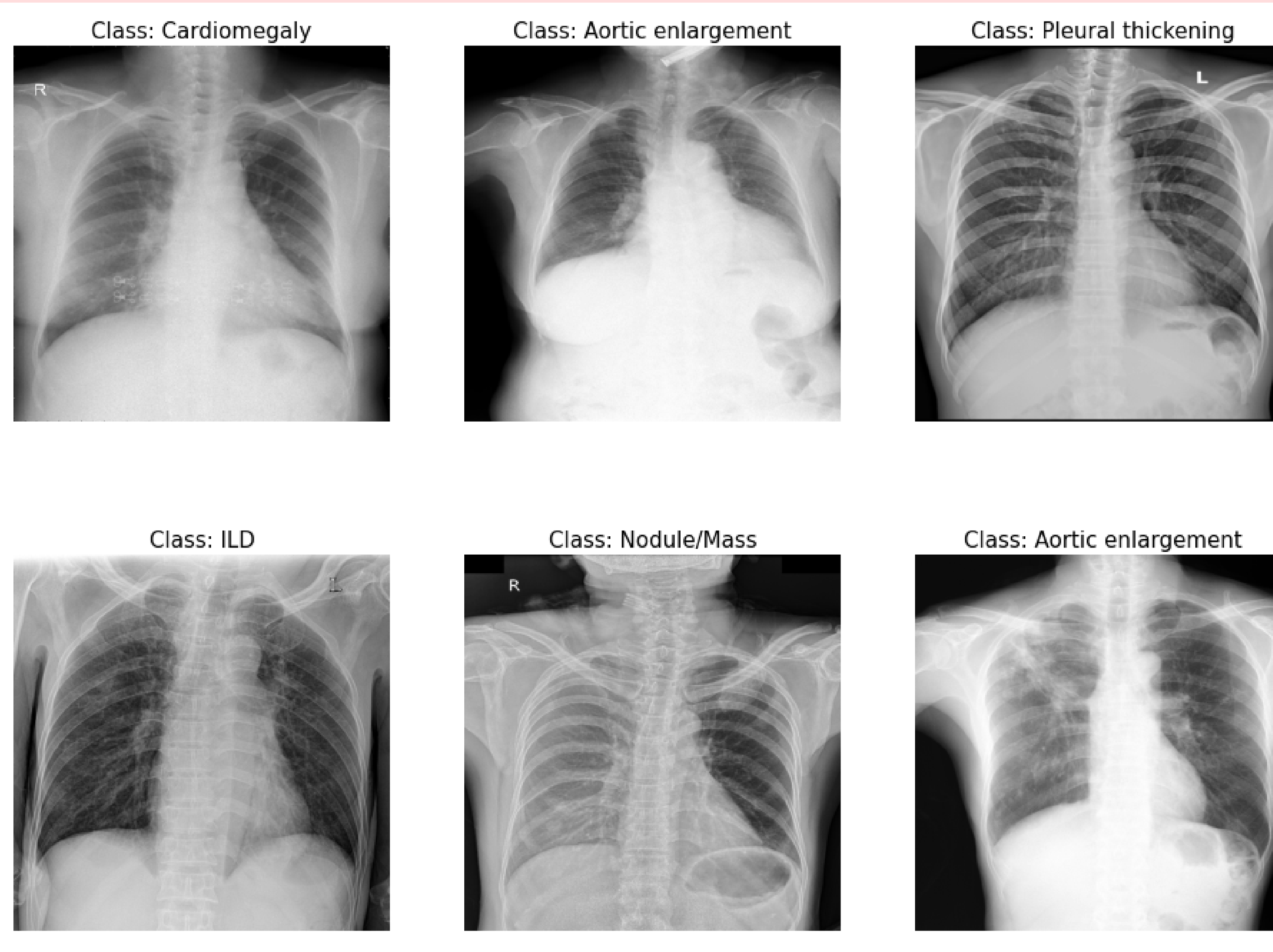
In [4]:
```
def dicom_to_np(path, voi_lut = True, fix_monochrome = True):
    dicom = pydicom.read_file(path)

    # VOI LUT (if available by DICOM device) is used to transform raw DICOM data to "human-friendly"
 view
    if voi_lut:
        data = apply_voi_lut(dicom.pixel_array, dicom)
    else:
        data = dicom.pixel_array

    # depending on this value, X-ray may look inverted - fix that:
    if fix_monochrome and dicom.PhotometricInterpretation == "MONOCHROME1":
        data = np.amax(data) - data

    data = data - np.min(data)
    data = data / np.max(data)
    data = (data * 255).astype(np.uint8)

    return data
```

Show images with class label

In [5]:
```



def show_img(img_ids, img_classes):
    plt.figure(figsize=(16, 12))

    for i, (img_id, img_class) in enumerate(zip(img_ids, img_classes)):
        plt.subplot(2, 3, i + 1)
        src= train_dir+ '/'+img_id
        img = dicom_to_np(src)
        img = cv2.resize(img, (500,500))
        plt.imshow(img, cmap='gray')
        plt.title(f"Class: {img_class}", fontsize=15)
        plt.axis("off")

    plt.show()
```

In [6]:
```
df = new_df[0:6]
img_ids = df["image_id"].values+'.dicom'
class_names = df["class_name"].values

show_img(img_ids, class_names)
```

```
/opt/conda/lib/python3.7/site-packages/pydicom/pixel_data_handlers/pillow_handler.py:177: UserWarnin
g: The (0028,0101) 'Bits Stored' value (12-bit) doesn't match the JPEG 2000 data (16-bit). It's recom
mended that you change the 'Bits Stored' value
  f"The (0028,0101) 'Bits Stored' value ({ds.BitsStored}-bit) "
```

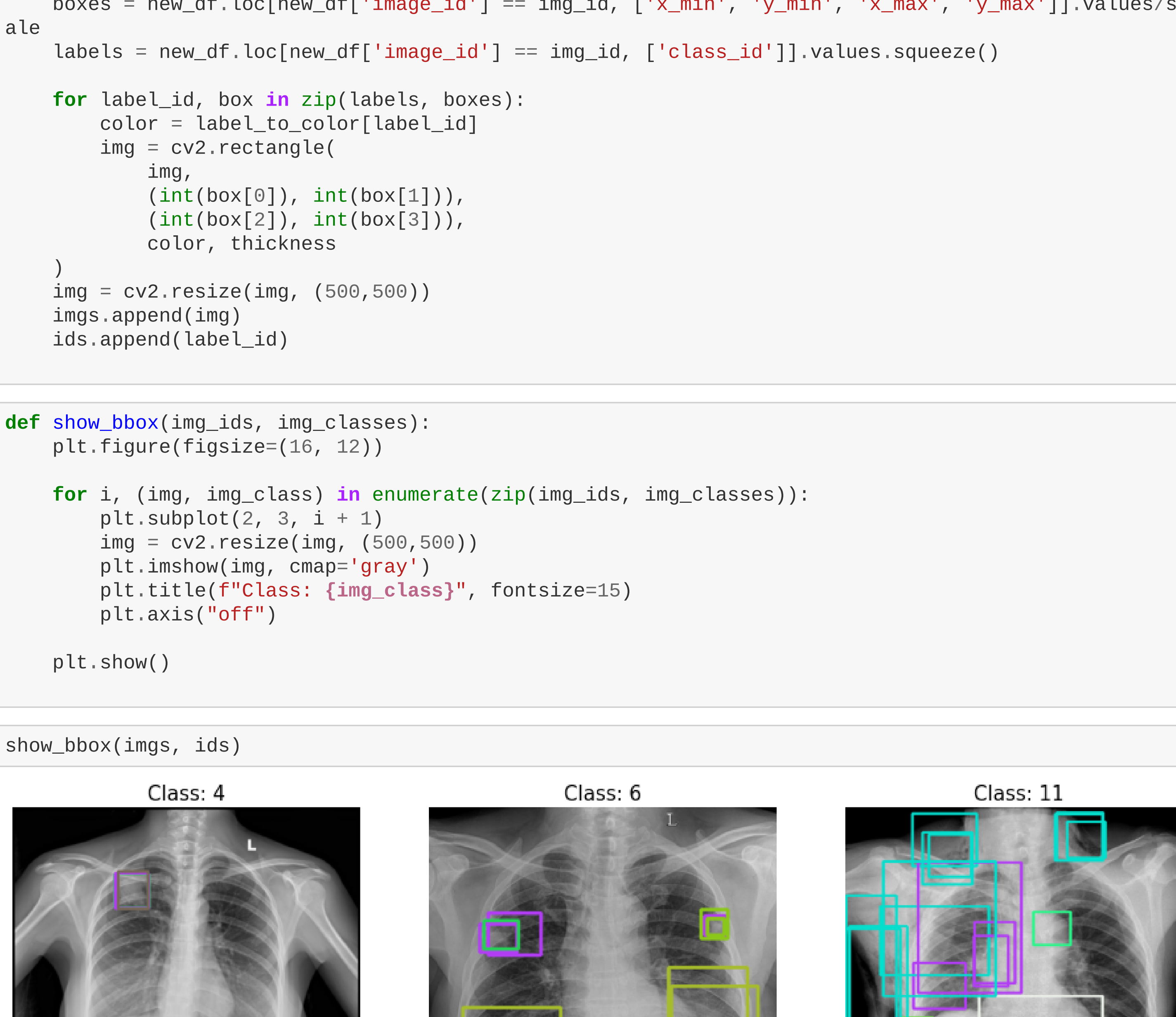


Draw bounding boxes

In [7]:
```
imgs = []
ids = []
img_ids = new_df['image_id'].values
class_ids = new_df['class_id'].unique()

label_to_color = {class_id:[randint(0,255) for i in range(3)] for class_id in class_ids}
thickness = 3
scale = 5


for i in range(6):
    img_id = random.choice(img_ids)
    img_path = f'{train_dir}/{img_id}.dicom'
    img = dicom_to_np(path=img_path)
    img = cv2.resize(img, None, fx=1/scale, fy=1/scale)
    img = np.stack([img, img, img], axis=-1)

    boxes = new_df.loc[new_df['image_id'] == img_id, ['x_min', 'y_min', 'x_max', 'y_max']].values/sc
ale
    labels = new_df.loc[new_df['image_id'] == img_id, ['class_id']].values.squeeze()

    for label_id, box in zip(labels, boxes):
        color = label_to_color[label_id]
        img = cv2.rectangle(
            img,
            (int(box[0]), int(box[1])),
            (int(box[2]), int(box[3])),
            color, thickness
    )
    img = cv2.resize(img, (500,500))
    imgs.append(img)
    ids.append(label_id)
```

In [8]:
```
def show_bbox(img_ids, img_classes):
    plt.figure(figsize=(16, 12))

    for i, (img, img_class) in enumerate(zip(img_ids, img_classes)):
        plt.subplot(2, 3, i + 1)
        img = cv2.resize(img, (500,500))
        plt.imshow(img, cmap='gray')
        plt.title(f"Class: {img_class}", fontsize=15)
        plt.axis("off")

    plt.show()
```

In [9]:
```
show_bbox(imgs, ids)
```



CSV File

- **image_id** - unique image identifier
- **class_name** - the name of the class of detected object (or "No finding")
- **class_id** - the ID of the class of detected object
- **rad_id** - the ID of the radiologist that made the observation
- **x_min** - minimum X coordinate of the object's bounding box
- **y_min** - minimum Y coordinate of the object's bounding box
- **x_max** - maximum X coordinate of the object's bounding box
- **y_max** - maximum Y coordinate of the object's bounding box

In [10]:
```
train_df.head()
```

Out[10]:

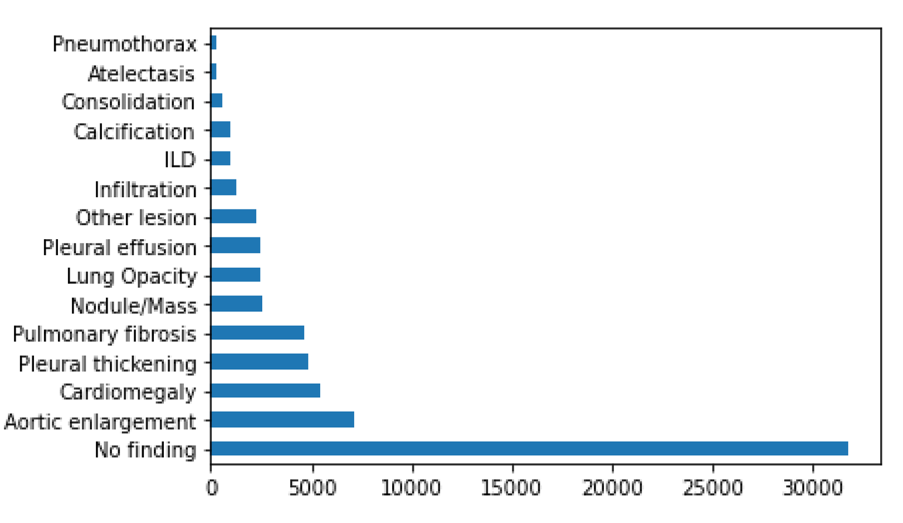
	image_id	class_name	class_id	rad_id	x_min	y_min	x_max	y_max
0	50a418190bc3fb1ef1633bf9678929b3	No finding	14	R11	NaN	NaN	NaN	NaN
1	21a10246a5ec7af151081d0cd6d65dc9	No finding	14	R7	NaN	NaN	NaN	NaN
2	9a5094b2563a1ef3ff50dc5c7ff71345	Cardiomegaly	3	R10	691.0	1375.0	1653.0	1831.0
3	051132a778e61a86eb147c7c6f564dfe	Aortic enlargement	0	R10	1264.0	743.0	1611.0	1019.0
4	063319de25ce7edb9b1c6b8881290140	No finding	14	R10	NaN	NaN	NaN	NaN

Visualize amount of value

Visualize value of every class with no founding.

In [11]:
```
train_df['class_name'].value_counts().plot.barh()
```

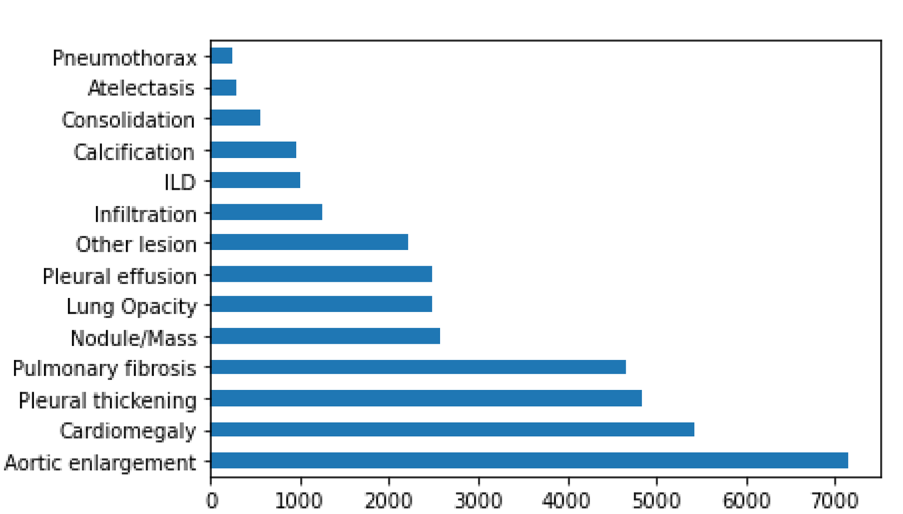
Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x7f1f8e141e50>



Visualize value of every class without no founding.

In [12]:
```
train_df.loc[train_df['class_name'] != 'No finding','class_name'].value_counts().plot.barh()
```

Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x7f1f8e093750>



Up vote Please.....

Working progress .