Evaluating Diagnostic Models

January 5, 2024

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

import util
from public_tests import *
from test_utils import *
```

In [5]:
```
train_results = pd.read_csv("data/train_preds.csv")
valid_results = pd.read_csv("data/valid_preds.csv")

# the labels in our dataset
class_labels = ['Cardiomegaly',
 'Emphysema',
 'Effusion',
 'Hernia',
 'Infiltration',
 'Mass',
 'Nodule',
 'Atelectasis',
 'Pneumothorax',
 'Pleural_Thickening',
 'Pneumonia',
 'Fibrosis',
 'Edema',
 'Consolidation']

# the labels for prediction values in our dataset
pred_labels = [l + "_pred" for l in class_labels]
```

In [6]:
```
y = valid_results[class_labels].values
pred = valid_results[pred_labels].values
```

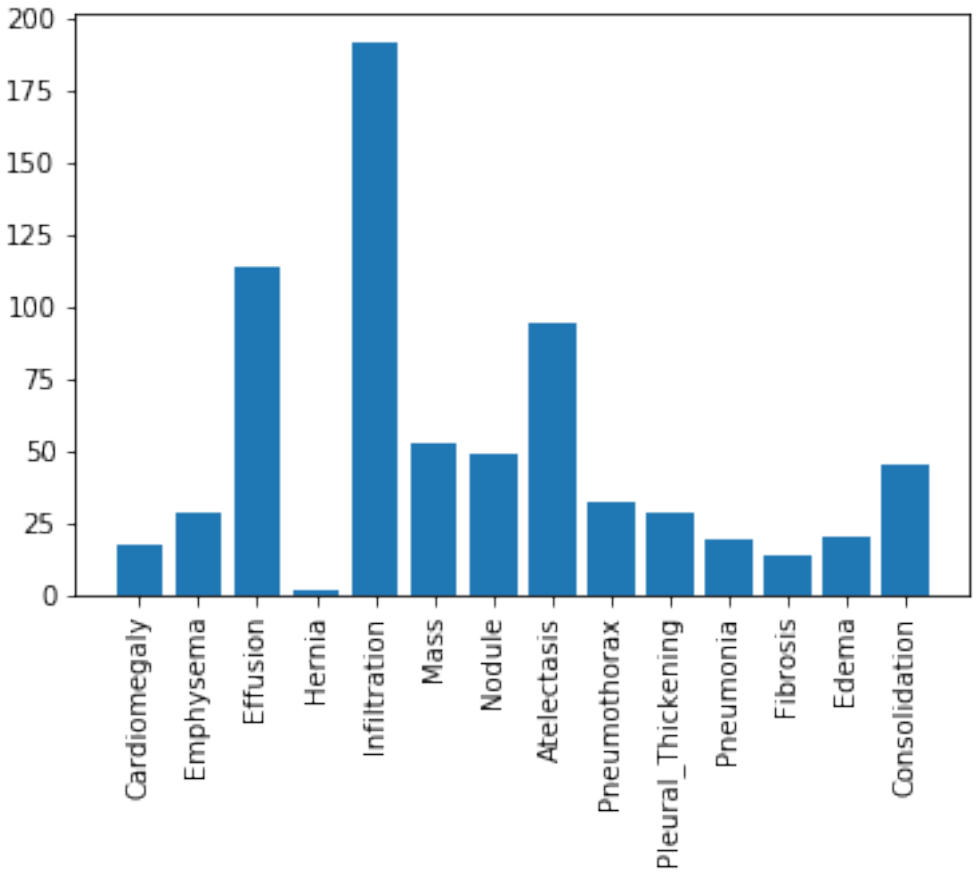
In [7]:
```
# let's take a peek at our dataset
valid_results[np.concatenate([class_labels, pred_labels])].head()
```

Out[7]:

	Cardiomegaly	Emphysema	Effusion	Hernia	Infiltration	Mass	Nodule	\
0	0	0	0	0	0	0	0	

```
1             0          0         0       0             1     0       1
2             0          0         0       0             0     0       0
3             0          0         0       0             0     0       0
4             0          0         0       0             0     0       0

   Atelectasis  Pneumothorax  Pleural_Thickening  ...  Infiltration_pred  \
0            0             0                   0  ...           0.256020
1            0             0                   0  ...           0.382199
2            0             0                   0  ...           0.427727
3            0             0                   0  ...           0.158596
4            0             0                   0  ...           0.536762

   Mass_pred  Nodule_pred  Atelectasis_pred  Pneumothorax_pred  \
0   0.266928     0.312440          0.460342           0.079453
1   0.176825     0.465807          0.489424           0.084595
2   0.115513     0.249030          0.035105           0.238761
3   0.259460     0.334870          0.266489           0.073371
4   0.198797     0.273110          0.186771           0.242122

   Pleural_Thickening_pred  Pneumonia_pred  Fibrosis_pred  Edema_pred  \
0                 0.271495        0.276861       0.398799    0.015867
1                 0.377318        0.363582       0.638024    0.025948
2                 0.167095        0.166389       0.262463    0.007758
3                 0.229834        0.191281       0.344348    0.008559
4                 0.309786        0.411771       0.244666    0.126930

   Consolidation_pred
0            0.156320
1            0.144419
2            0.125790
3            0.119153
4            0.342409

[5 rows x 28 columns]
```

In [8]:
```
plt.xticks(rotation=90)
plt.bar(x = class_labels, height= y.sum(axis=0));
```




```
In [9]: # UNQ_C1 (UNIQUE CELL IDENTIFIER, DO NOT EDIT)
        def true_positives(y, pred, th=0.5):
            """
            Count true positives.

            Args:
                y (np.array): ground truth, size (n_examples)
                pred (np.array): model output, size (n_examples)
                th (float): cutoff value for positive prediction from model
            Returns:
                TP (int): true positives
            """
            TP = 0

            # get thresholded predictions
            thresholded_preds = pred >= th

            # compute TP
            TP = np.sum((y == 1) & (thresholded_preds == 1))
```

```
    return TP

def true_negatives(y, pred, th=0.5):
    """
    Count true negatives.

    Args:
        y (np.array): ground truth, size (n_examples)
        pred (np.array): model output, size (n_examples)
        th (float): cutoff value for positive prediction from model
    Returns:
        TN (int): true negatives
    """
    TN = 0

    # get thresholded predictions
    thresholded_preds = pred >= th

    # compute TN
    TN = np.sum((y == 0) & (thresholded_preds == 0))

    return TN

def false_positives(y, pred, th=0.5):
    """
    Count false positives.

    Args:
        y (np.array): ground truth, size (n_examples)
        pred (np.array): model output, size (n_examples)
        th (float): cutoff value for positive prediction from model
    Returns:
        FP (int): false positives
    """
    FP = 0

    # get thresholded predictions
    thresholded_preds = pred >= th

    # compute FP
    FP = np.sum((y == 0) & (thresholded_preds == 1))

    return FP

def false_negatives(y, pred, th=0.5):
    """
    Count false positives.
```

```
            Args:
                y (np.array): ground truth, size (n_examples)
                pred (np.array): model output, size (n_examples)
                th (float): cutoff value for positive prediction from model
            Returns:
                FN (int): false negatives
            """
            FN = 0

            # get thresholded predictions
            thresholded_preds = pred >= th

            # compute FN
            FN = np.sum((y == 1) & (thresholded_preds == 0))

            return FN
```

In [10]:
```
### do not modify this cell
get_tp_tn_fp_fn_test(true_positives, true_negatives, false_positives, false_negatives)
```

	y_test	preds_test	category
0	1	0.8	TP
1	1	0.7	TP
2	0	0.4	TN
3	0	0.3	TN
4	0	0.2	TN
5	0	0.5	FP
6	0	0.6	FP
7	0	0.7	FP
8	0	0.8	FP
9	1	0.1	FN
10	1	0.2	FN
11	1	0.3	FN
12	1	0.4	FN
13	1	0.0	FN

```
Your functions calcualted:
    TP: 2
    TN: 3
    FP: 4
    FN: 5
```

All tests passed. All tests passed. All tests passed. All tests passed.

In [ ]:
```
util.get_performance_metrics(y, pred, class_labels)
```

In [11]:
```
# UNQ_C2 (UNIQUE CELL IDENTIFIER, DO NOT EDIT)
def get_accuracy(y, pred, th=0.5):
```

```
    """
    Compute accuracy of predictions at threshold.

    Args:
        y (np.array): ground truth, size (n_examples)
        pred (np.array): model output, size (n_examples)
        th (float): cutoff value for positive prediction from model
    Returns:
        accuracy (float): accuracy of predictions at threshold
    """
    accuracy = 0.0

    ### START CODE HERE (REPLACE INSTANCES OF 'None' with your code) ###

    # get TP, FP, TN, FN using our previously defined functions
    TP = true_positives(y, pred, th)
    FP = false_positives(y, pred, th)
    TN = true_negatives(y, pred, th)
    FN = false_negatives(y, pred, th)

    # Compute accuracy using TP, FP, TN, FN
    accuracy = (TP + TN) / (TP + TN + FP + FN) if (TP + TN + FP + FN) != 0 else 0.0

    ### END CODE HERE ###

    return accuracy
```

In [12]:
```
### do not modify this cell
get_accuracy_test(get_accuracy)
```

```
Test Case:

Test Labels:           [1 0 0 1 1]
Test Predictions:  [0.8 0.8 0.4 0.6 0.3]
Threshold:           0.5
Computed Accuracy: 0.6

 All tests passed.
```

In [13]:
```
util.get_performance_metrics(y, pred, class_labels, acc=get_accuracy)
```

Out[13]:

	TP	TN	FP	FN	Accuracy	Prevalence	Sensitivity	\
Cardiomegaly	16	814	169	1	0.83	Not Defined	Not Defined	
Emphysema	20	869	103	8	0.889	Not Defined	Not Defined	
Effusion	99	690	196	15	0.789	Not Defined	Not Defined	
Hernia	1	743	255	1	0.744	Not Defined	Not Defined	
Infiltration	114	543	265	78	0.657	Not Defined	Not Defined	
Mass	40	789	158	13	0.829	Not Defined	Not Defined	

```
                    28  731  220  21    0.759  Not Defined  Not Defined
Atelectasis         64  657  249  30    0.721  Not Defined  Not Defined
Pneumothorax        24  785  183   8    0.809  Not Defined  Not Defined
Pleural_Thickening  24  713  259   4    0.737  Not Defined  Not Defined
Pneumonia           14  661  320   5    0.675  Not Defined  Not Defined
Fibrosis            10  725  261   4    0.735  Not Defined  Not Defined
Edema               15  767  213   5    0.782  Not Defined  Not Defined
Consolidation       36  658  297   9    0.694  Not Defined  Not Defined

                    Specificity          PPV          NPV          AUC  \

Cardiomegaly        Not Defined  Not Defined  Not Defined  Not Defined
Emphysema           Not Defined  Not Defined  Not Defined  Not Defined
Effusion            Not Defined  Not Defined  Not Defined  Not Defined
Hernia              Not Defined  Not Defined  Not Defined  Not Defined
Infiltration        Not Defined  Not Defined  Not Defined  Not Defined
Mass                Not Defined  Not Defined  Not Defined  Not Defined
Nodule              Not Defined  Not Defined  Not Defined  Not Defined
Atelectasis         Not Defined  Not Defined  Not Defined  Not Defined
Pneumothorax        Not Defined  Not Defined  Not Defined  Not Defined
Pleural_Thickening  Not Defined  Not Defined  Not Defined  Not Defined
Pneumonia           Not Defined  Not Defined  Not Defined  Not Defined
Fibrosis            Not Defined  Not Defined  Not Defined  Not Defined
Edema               Not Defined  Not Defined  Not Defined  Not Defined
Consolidation       Not Defined  Not Defined  Not Defined  Not Defined

                              F1 Threshold

Cardiomegaly        Not Defined       0.5
Emphysema           Not Defined       0.5
Effusion            Not Defined       0.5
Hernia              Not Defined       0.5
Infiltration        Not Defined       0.5
Mass                Not Defined       0.5
Nodule              Not Defined       0.5
Atelectasis         Not Defined       0.5
Pneumothorax        Not Defined       0.5
Pleural_Thickening  Not Defined       0.5
Pneumonia           Not Defined       0.5
Fibrosis            Not Defined       0.5
Edema               Not Defined       0.5
Consolidation       Not Defined       0.5
```

In [14]:
```
get_accuracy(valid_results["Emphysema"].values, np.zeros(len(valid_results)))
```

Out[14]: 0.972

In [15]:
```
# UNQ_C3 (UNIQUE CELL IDENTIFIER, DO NOT EDIT)
def get_prevalence(y):
```

```
    """
    Compute prevalence.

    Args:
        y (np.array): ground truth, size (n_examples)
    Returns:
        prevalence (float): prevalence of positive cases
    """
    prevalence = 0.0

    ### START CODE HERE (REPLACE INSTANCES OF 'None' with your code) ###

    prevalence = np.mean(y)

    ### END CODE HERE ###

    return prevalence
```

In [16]:
```
### do npt modify this cell
get_prevalence_test(get_prevalence)
```

```
Test Case:

Test Labels:              [1 0 0 1 1 0 0 0 0 1]
Computed Prevalence:  0.4

 All tests passed.
```

In [17]:
```
util.get_performance_metrics(y, pred, class_labels, acc=get_accuracy, prevalence=get_
```

Out[17]:

	TP	TN	FP	FN	Accuracy	Prevalence	Sensitivity	\
Cardiomegaly	16	814	169	1	0.83	0.017	Not Defined	
Emphysema	20	869	103	8	0.889	0.028	Not Defined	
Effusion	99	690	196	15	0.789	0.114	Not Defined	
Hernia	1	743	255	1	0.744	0.002	Not Defined	
Infiltration	114	543	265	78	0.657	0.192	Not Defined	
Mass	40	789	158	13	0.829	0.053	Not Defined	
Nodule	28	731	220	21	0.759	0.049	Not Defined	
Atelectasis	64	657	249	30	0.721	0.094	Not Defined	
Pneumothorax	24	785	183	8	0.809	0.032	Not Defined	
Pleural_Thickening	24	713	259	4	0.737	0.028	Not Defined	
Pneumonia	14	661	320	5	0.675	0.019	Not Defined	
Fibrosis	10	725	261	4	0.735	0.014	Not Defined	
Edema	15	767	213	5	0.782	0.02	Not Defined	
Consolidation	36	658	297	9	0.694	0.045	Not Defined	

	Specificity	PPV	NPV	AUC	\

```
Cardiomegaly        Not Defined  Not Defined  Not Defined  Not Defined
Emphysema           Not Defined  Not Defined  Not Defined  Not Defined
Effusion            Not Defined  Not Defined  Not Defined  Not Defined
Hernia              Not Defined  Not Defined  Not Defined  Not Defined
Infiltration        Not Defined  Not Defined  Not Defined  Not Defined
Mass                Not Defined  Not Defined  Not Defined  Not Defined
Nodule              Not Defined  Not Defined  Not Defined  Not Defined
Atelectasis         Not Defined  Not Defined  Not Defined  Not Defined
Pneumothorax        Not Defined  Not Defined  Not Defined  Not Defined
Pleural_Thickening  Not Defined  Not Defined  Not Defined  Not Defined
Pneumonia           Not Defined  Not Defined  Not Defined  Not Defined
Fibrosis            Not Defined  Not Defined  Not Defined  Not Defined
Edema               Not Defined  Not Defined  Not Defined  Not Defined
Consolidation       Not Defined  Not Defined  Not Defined  Not Defined

                                 F1 Threshold

Cardiomegaly        Not Defined       0.5
Emphysema           Not Defined       0.5
Effusion            Not Defined       0.5
Hernia              Not Defined       0.5
Infiltration        Not Defined       0.5
Mass                Not Defined       0.5
Nodule              Not Defined       0.5
Atelectasis         Not Defined       0.5
Pneumothorax        Not Defined       0.5
Pleural_Thickening  Not Defined       0.5
Pneumonia           Not Defined       0.5
Fibrosis            Not Defined       0.5
Edema               Not Defined       0.5
Consolidation       Not Defined       0.5
```

In [18]:
```python
# UNQ_C4 (UNIQUE CELL IDENTIFIER, DO NOT EDIT)
def get_sensitivity(y, pred, th=0.5):
    """
    Compute sensitivity of predictions at threshold.

    Args:
        y (np.array): ground truth, size (n_examples)
        pred (np.array): model output, size (n_examples)
        th (float): cutoff value for positive prediction from model
    Returns:
        sensitivity (float): probability that our test outputs positive given that th
    """
    sensitivity = 0.0

    ### START CODE HERE (REPLACE INSTANCES OF 'None' with your code) ###
```

```
            # get TP and FN using our previously defined functions
            TP = true_positives(y, pred, th)
            FN = false_negatives(y, pred, th)

            # use TP and FN to compute sensitivity
            sensitivity = TP / (TP + FN) if (TP + FN) != 0 else 0.0

            ### END CODE HERE ###

            return sensitivity

        def get_specificity(y, pred, th=0.5):
            """
            Compute specificity of predictions at threshold.

            Args:
                y (np.array): ground truth, size (n_examples)
                pred (np.array): model output, size (n_examples)
                th (float): cutoff value for positive prediction from model
            Returns:
                specificity (float): probability that the test outputs negative given that th
            """
            specificity = 0.0

            ### START CODE HERE (REPLACE INSTANCES OF 'None' with your code) ###

            # get TN and FP using our previously defined functions
            TN = true_negatives(y, pred, th)
            FP = false_positives(y, pred, th)

            # use TN and FP to compute specificity
            specificity = TN / (TN + FP) if (TN + FP) != 0 else 0.0

            ### END CODE HERE ###

            return specificity
```

In [19]:
```
### do not modify this cell
get_sensitivity_specificity_test(get_sensitivity, get_specificity)
```

```
Test Case:

Test Labels:                [1 0 0 1 1]
Test Predictions:       [1 0 0 1 1]
Threshold:                  0.5
Computed Sensitivity:  0.6666666666666666
Computed Specificity:  0.5

 All tests passed. All tests passed.
```

In [20]:
```
util.get_performance_metrics(y, pred, class_labels, acc=get_accuracy, prevalence=get_
                             sens=get_sensitivity, spec=get_specificity)
```

Out[20]:

	TP	TN	FP	FN	Accuracy	Prevalence	Sensitivity	\
Cardiomegaly	16	814	169	1	0.83	0.017	0.941	
Emphysema	20	869	103	8	0.889	0.028	0.714	
Effusion	99	690	196	15	0.789	0.114	0.868	
Hernia	1	743	255	1	0.744	0.002	0.5	
Infiltration	114	543	265	78	0.657	0.192	0.594	
Mass	40	789	158	13	0.829	0.053	0.755	
Nodule	28	731	220	21	0.759	0.049	0.571	
Atelectasis	64	657	249	30	0.721	0.094	0.681	
Pneumothorax	24	785	183	8	0.809	0.032	0.75	
Pleural_Thickening	24	713	259	4	0.737	0.028	0.857	
Pneumonia	14	661	320	5	0.675	0.019	0.737	
Fibrosis	10	725	261	4	0.735	0.014	0.714	
Edema	15	767	213	5	0.782	0.02	0.75	
Consolidation	36	658	297	9	0.694	0.045	0.8	

	Specificity	PPV	NPV	AUC	\
Cardiomegaly	0.828	Not Defined	Not Defined	Not Defined	
Emphysema	0.894	Not Defined	Not Defined	Not Defined	
Effusion	0.779	Not Defined	Not Defined	Not Defined	
Hernia	0.744	Not Defined	Not Defined	Not Defined	
Infiltration	0.672	Not Defined	Not Defined	Not Defined	
Mass	0.833	Not Defined	Not Defined	Not Defined	
Nodule	0.769	Not Defined	Not Defined	Not Defined	
Atelectasis	0.725	Not Defined	Not Defined	Not Defined	
Pneumothorax	0.811	Not Defined	Not Defined	Not Defined	
Pleural_Thickening	0.734	Not Defined	Not Defined	Not Defined	
Pneumonia	0.674	Not Defined	Not Defined	Not Defined	
Fibrosis	0.735	Not Defined	Not Defined	Not Defined	
Edema	0.783	Not Defined	Not Defined	Not Defined	
Consolidation	0.689	Not Defined	Not Defined	Not Defined	

	F1	Threshold
Cardiomegaly	Not Defined	0.5
Emphysema	Not Defined	0.5
Effusion	Not Defined	0.5
Hernia	Not Defined	0.5
Infiltration	Not Defined	0.5
Mass	Not Defined	0.5
Nodule	Not Defined	0.5
Atelectasis	Not Defined	0.5
Pneumothorax	Not Defined	0.5

```
Pleural_Thickening  Not Defined       0.5
Pneumonia           Not Defined       0.5
Fibrosis            Not Defined       0.5
Edema               Not Defined       0.5
Consolidation       Not Defined       0.5
```

In [21]:
```python
# UNQ_C5 (UNIQUE CELL IDENTIFIER, DO NOT EDIT)
def get_ppv(y, pred, th=0.5):
    """
    Compute PPV of predictions at threshold.

    Args:
        y (np.array): ground truth, size (n_examples)
        pred (np.array): model output, size (n_examples)
        th (float): cutoff value for positive prediction from model
    Returns:
        PPV (float): positive predictive value of predictions at threshold
    """
    PPV = 0.0

    ### START CODE HERE (REPLACE INSTANCES OF 'None' with your code) ###

    # get TP and FP using our previously defined functions
    TP = true_positives(y, pred, th)
    FP = false_positives(y, pred, th)

    # use TP and FP to compute PPV
    PPV = TP / (TP + FP) if (TP + FP) != 0 else 0.0

    ### END CODE HERE ###

    return PPV

def get_npv(y, pred, th=0.5):
    """
    Compute NPV of predictions at threshold.

    Args:
        y (np.array): ground truth, size (n_examples)
        pred (np.array): model output, size (n_examples)
        th (float): cutoff value for positive prediction from model
    Returns:
        NPV (float): negative predictive value of predictions at threshold
    """
    NPV = 0.0

    ### START CODE HERE (REPLACE INSTANCES OF 'None' with your code) ###
```

```
    # get TN and FN using our previously defined functions
    TN = true_negatives(y, pred, th)
    FN = false_negatives(y, pred, th)

    # use TN and FN to compute NPV
    NPV = TN / (TN + FN) if (TN + FN) != 0 else 0.0

    ### END CODE HERE ###

    return NPV
```

In [22]:
```
### do not modify this cell
get_ppv_npv_test(get_ppv, get_npv)
```

```
Test Case:

Test Labels:           [1 0 0 1 1]
Test Predictions:  [1 0 0 1 1]
Threshold:           0.5
Computed PPV:           0.6666666666666666
Computed NPV:           0.5

 All tests passed. All tests passed.
```

In [23]:
```
util.get_performance_metrics(y, pred, class_labels, acc=get_accuracy, prevalence=get_
                             sens=get_sensitivity, spec=get_specificity, ppv=get_ppv, npv=
```

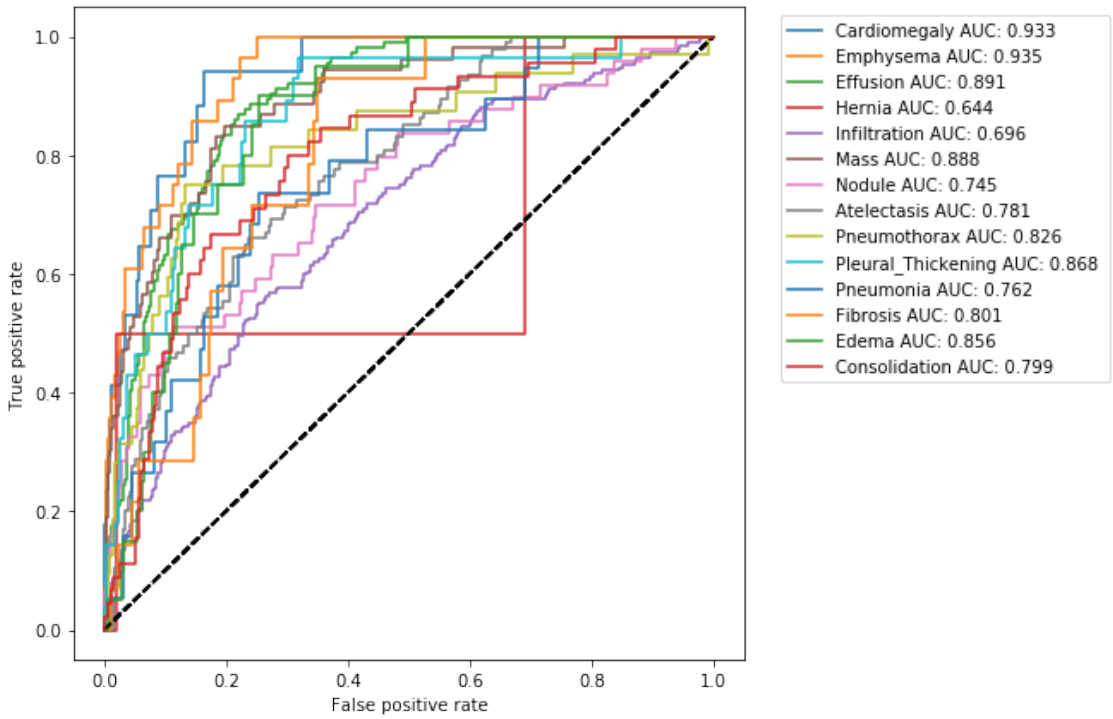
Out[23]:

	TP	TN	FP	FN	Accuracy	Prevalence	Sensitivity	\
Cardiomegaly	16	814	169	1	0.83	0.017	0.941	
Emphysema	20	869	103	8	0.889	0.028	0.714	
Effusion	99	690	196	15	0.789	0.114	0.868	
Hernia	1	743	255	1	0.744	0.002	0.5	
Infiltration	114	543	265	78	0.657	0.192	0.594	
Mass	40	789	158	13	0.829	0.053	0.755	
Nodule	28	731	220	21	0.759	0.049	0.571	
Atelectasis	64	657	249	30	0.721	0.094	0.681	
Pneumothorax	24	785	183	8	0.809	0.032	0.75	
Pleural_Thickening	24	713	259	4	0.737	0.028	0.857	
Pneumonia	14	661	320	5	0.675	0.019	0.737	
Fibrosis	10	725	261	4	0.735	0.014	0.714	
Edema	15	767	213	5	0.782	0.02	0.75	
Consolidation	36	658	297	9	0.694	0.045	0.8	

	Specificity	PPV	NPV	AUC	F1	\
Cardiomegaly	0.828	0.086	0.999	Not Defined	Not Defined	
Emphysema	0.894	0.163	0.991	Not Defined	Not Defined	
Effusion	0.779	0.336	0.979	Not Defined	Not Defined	

```
Hernia                  0.744  0.004  0.999  Not Defined  Not Defined
Infiltration            0.672  0.301  0.874  Not Defined  Not Defined
Mass                    0.833  0.202  0.984  Not Defined  Not Defined
Nodule                  0.769  0.113  0.972  Not Defined  Not Defined
Atelectasis             0.725  0.204  0.956  Not Defined  Not Defined
Pneumothorax            0.811  0.116   0.99  Not Defined  Not Defined
Pleural_Thickening      0.734  0.085  0.994  Not Defined  Not Defined
Pneumonia               0.674  0.042  0.992  Not Defined  Not Defined
Fibrosis                0.735  0.037  0.995  Not Defined  Not Defined
Edema                   0.783  0.066  0.994  Not Defined  Not Defined
Consolidation           0.689  0.108  0.987  Not Defined  Not Defined

                    Threshold

Cardiomegaly              0.5
Emphysema                 0.5
Effusion                  0.5
Hernia                    0.5
Infiltration              0.5
Mass                      0.5
Nodule                    0.5
Atelectasis               0.5
Pneumothorax              0.5
Pleural_Thickening        0.5
Pneumonia                 0.5
Fibrosis                  0.5
Edema                     0.5
Consolidation             0.5
```

In [24]:
```
util.get_curve(y, pred, class_labels)
```




```
In [25]: from sklearn.metrics import roc_auc_score
         util.get_performance_metrics(y, pred, class_labels, acc=get_accuracy, prevalence=get_
                                      sens=get_sensitivity, spec=get_specificity, ppv=get_ppv, npv=
```

Out[25]:

	TP	TN	FP	FN	Accuracy	Prevalence	Sensitivity	\
Cardiomegaly	16	814	169	1	0.83	0.017	0.941	
Emphysema	20	869	103	8	0.889	0.028	0.714	
Effusion	99	690	196	15	0.789	0.114	0.868	
Hernia	1	743	255	1	0.744	0.002	0.5	
Infiltration	114	543	265	78	0.657	0.192	0.594	
Mass	40	789	158	13	0.829	0.053	0.755	
Nodule	28	731	220	21	0.759	0.049	0.571	
Atelectasis	64	657	249	30	0.721	0.094	0.681	
Pneumothorax	24	785	183	8	0.809	0.032	0.75	
Pleural_Thickening	24	713	259	4	0.737	0.028	0.857	
Pneumonia	14	661	320	5	0.675	0.019	0.737	
Fibrosis	10	725	261	4	0.735	0.014	0.714	
Edema	15	767	213	5	0.782	0.02	0.75	
Consolidation	36	658	297	9	0.694	0.045	0.8	

	Specificity	PPV	NPV	AUC	F1	Threshold
Cardiomegaly	0.828	0.086	0.999	0.933	Not Defined	0.5

```
Emphysema                0.894  0.163  0.991  0.935  Not Defined       0.5
Effusion                 0.779  0.336  0.979  0.891  Not Defined       0.5
Hernia                   0.744  0.004  0.999  0.644  Not Defined       0.5
Infiltration             0.672  0.301  0.874  0.696  Not Defined       0.5
Mass                     0.833  0.202  0.984  0.888  Not Defined       0.5
Nodule                   0.769  0.113  0.972  0.745  Not Defined       0.5
Atelectasis              0.725  0.204  0.956  0.781  Not Defined       0.5
Pneumothorax             0.811  0.116   0.99  0.826  Not Defined       0.5
Pleural_Thickening       0.734  0.085  0.994  0.868  Not Defined       0.5
Pneumonia                0.674  0.042  0.992  0.762  Not Defined       0.5
Fibrosis                 0.735  0.037  0.995  0.801  Not Defined       0.5
Edema                    0.783  0.066  0.994  0.856  Not Defined       0.5
Consolidation            0.689  0.108  0.987  0.799  Not Defined       0.5
```

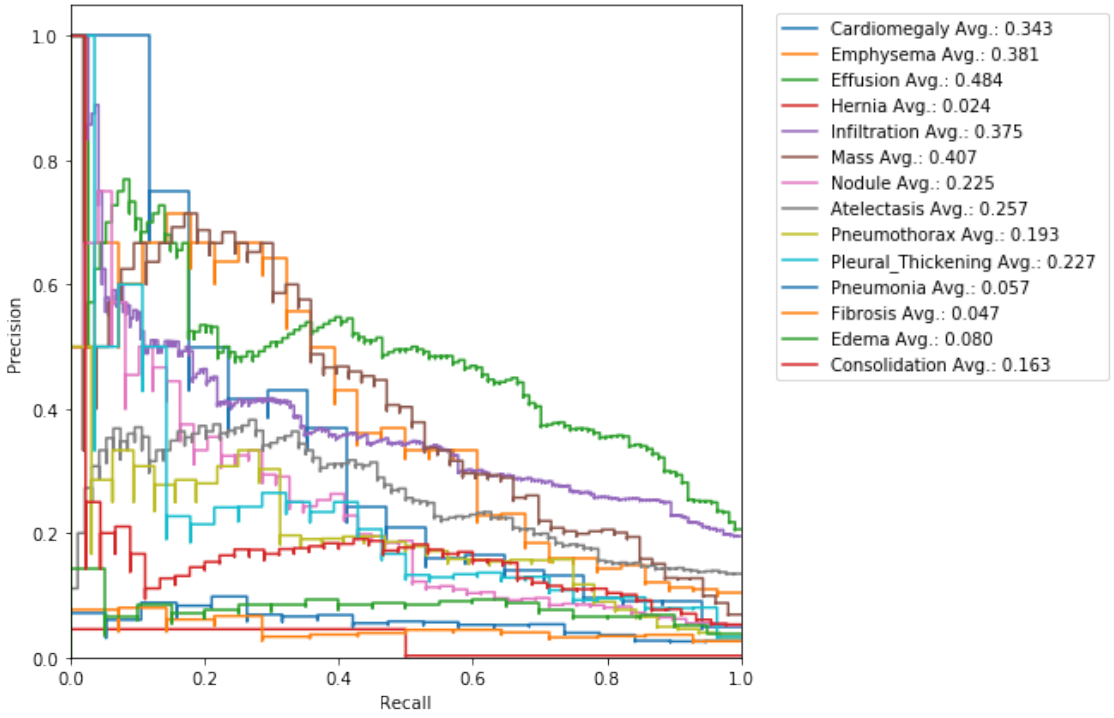
In [26]:
```
def bootstrap_auc(y, pred, classes, bootstraps = 100, fold_size = 1000):
    statistics = np.zeros((len(classes), bootstraps))

    for c in range(len(classes)):
        df = pd.DataFrame(columns=['y', 'pred'])
        df.loc[:, 'y'] = y[:, c]
        df.loc[:, 'pred'] = pred[:, c]
        # get positive examples for stratified sampling
        df_pos = df[df.y == 1]
        df_neg = df[df.y == 0]
        prevalence = len(df_pos) / len(df)
        for i in range(bootstraps):
            # stratified sampling of positive and negative examples
            pos_sample = df_pos.sample(n = int(fold_size * prevalence), replace=True)
            neg_sample = df_neg.sample(n = int(fold_size * (1-prevalence)), replace=T

            y_sample = np.concatenate([pos_sample.y.values, neg_sample.y.values])
            pred_sample = np.concatenate([pos_sample.pred.values, neg_sample.pred.valu
            score = roc_auc_score(y_sample, pred_sample)
            statistics[c][i] = score
    return statistics

statistics = bootstrap_auc(y, pred, class_labels)
```

In [27]:
```
util.print_confidence_intervals(class_labels, statistics)
```

Out[27]:
```
                  Mean AUC (CI 5%-95%)
Cardiomegaly          0.93 (0.91-0.97)
Emphysema             0.94 (0.90-0.96)
Effusion              0.89 (0.87-0.91)
Hernia                0.64 (0.29-0.98)
Infiltration          0.70 (0.66-0.73)
Mass                  0.89 (0.85-0.92)
Nodule                0.74 (0.69-0.81)
```

```
Atelectasis              0.78 (0.75-0.82)
Pneumothorax             0.83 (0.75-0.91)
Pleural_Thickening       0.86 (0.82-0.92)
Pneumonia                0.76 (0.67-0.83)
Fibrosis                 0.80 (0.74-0.86)
Edema                    0.86 (0.80-0.90)
Consolidation            0.80 (0.75-0.86)
```

In [28]:
```
util.get_curve(y, pred, class_labels, curve='prc')
```



In [29]:
```
from sklearn.metrics import f1_score
util.get_performance_metrics(y, pred, class_labels, acc=get_accuracy, prevalence=get_
                             sens=get_sensitivity, spec=get_specificity, ppv=get_ppv, npv=
```

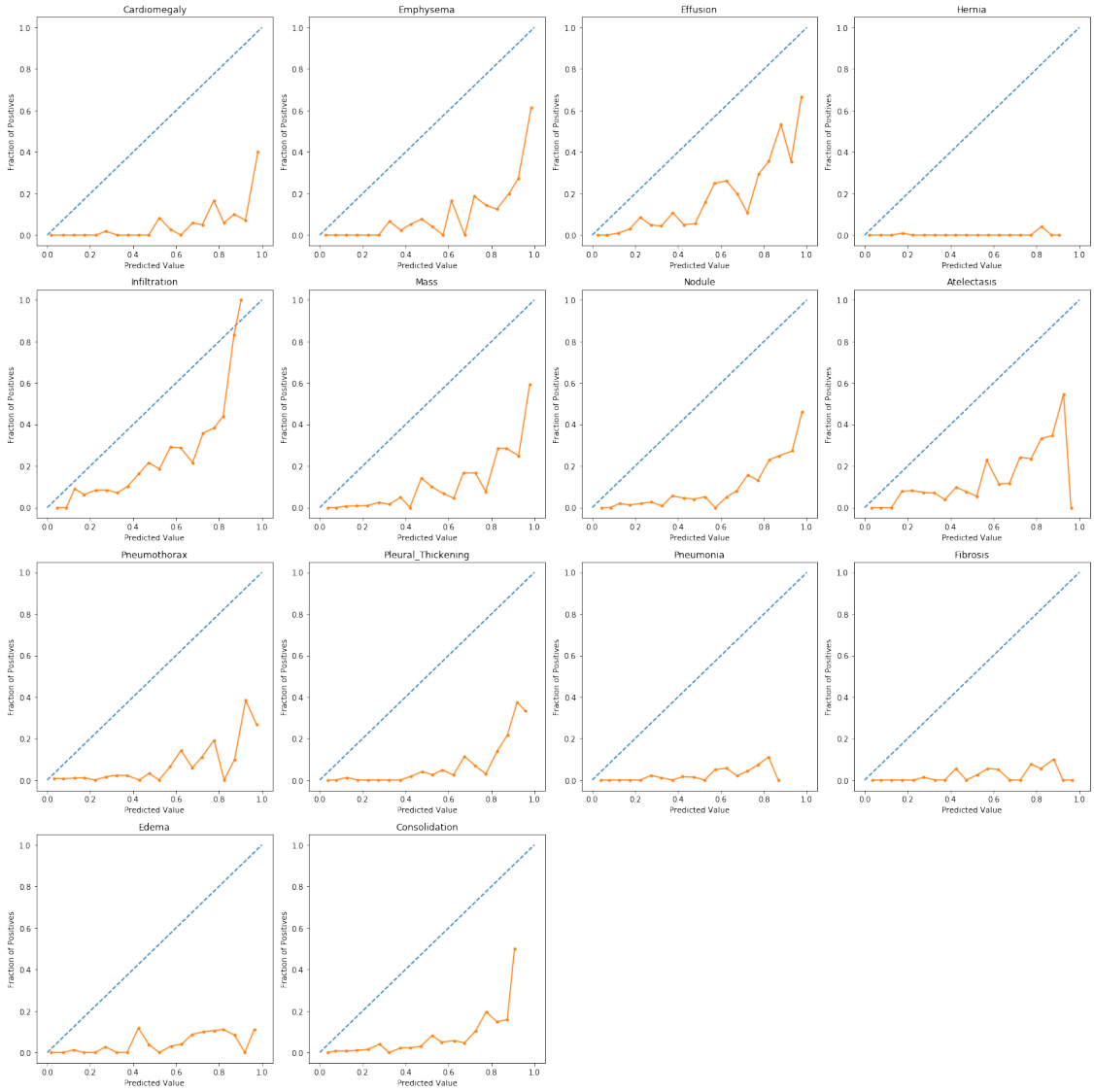
Out[29]:

	TP	TN	FP	FN	Accuracy	Prevalence	Sensitivity	\
Cardiomegaly	16	814	169	1	0.83	0.017	0.941	
Emphysema	20	869	103	8	0.889	0.028	0.714	
Effusion	99	690	196	15	0.789	0.114	0.868	
Hernia	1	743	255	1	0.744	0.002	0.5	
Infiltration	114	543	265	78	0.657	0.192	0.594	
Mass	40	789	158	13	0.829	0.053	0.755	
Nodule	28	731	220	21	0.759	0.049	0.571	
Atelectasis	64	657	249	30	0.721	0.094	0.681	

```
Pneumothorax          24  785  183   8    0.809      0.032        0.75
Pleural_Thickening    24  713  259   4    0.737      0.028       0.857
Pneumonia             14  661  320   5    0.675      0.019       0.737
Fibrosis              10  725  261   4    0.735      0.014       0.714
Edema                 15  767  213   5    0.782       0.02        0.75
Consolidation         36  658  297   9    0.694      0.045         0.8

                    Specificity    PPV    NPV    AUC     F1 Threshold

Cardiomegaly              0.828  0.086  0.999  0.933  0.158       0.5
Emphysema                 0.894  0.163  0.991  0.935  0.265       0.5
Effusion                  0.779  0.336  0.979  0.891  0.484       0.5
Hernia                    0.744  0.004  0.999  0.644  0.008       0.5
Infiltration              0.672  0.301  0.874  0.696  0.399       0.5
Mass                      0.833  0.202  0.984  0.888  0.319       0.5
Nodule                    0.769  0.113  0.972  0.745  0.189       0.5
Atelectasis               0.725  0.204  0.956  0.781  0.314       0.5
Pneumothorax              0.811  0.116   0.99  0.826  0.201       0.5
Pleural_Thickening        0.734  0.085  0.994  0.868  0.154       0.5
Pneumonia                 0.674  0.042  0.992  0.762  0.079       0.5
Fibrosis                  0.735  0.037  0.995  0.801   0.07       0.5
Edema                     0.783  0.066  0.994  0.856  0.121       0.5
Consolidation             0.689  0.108  0.987  0.799   0.19       0.5
```

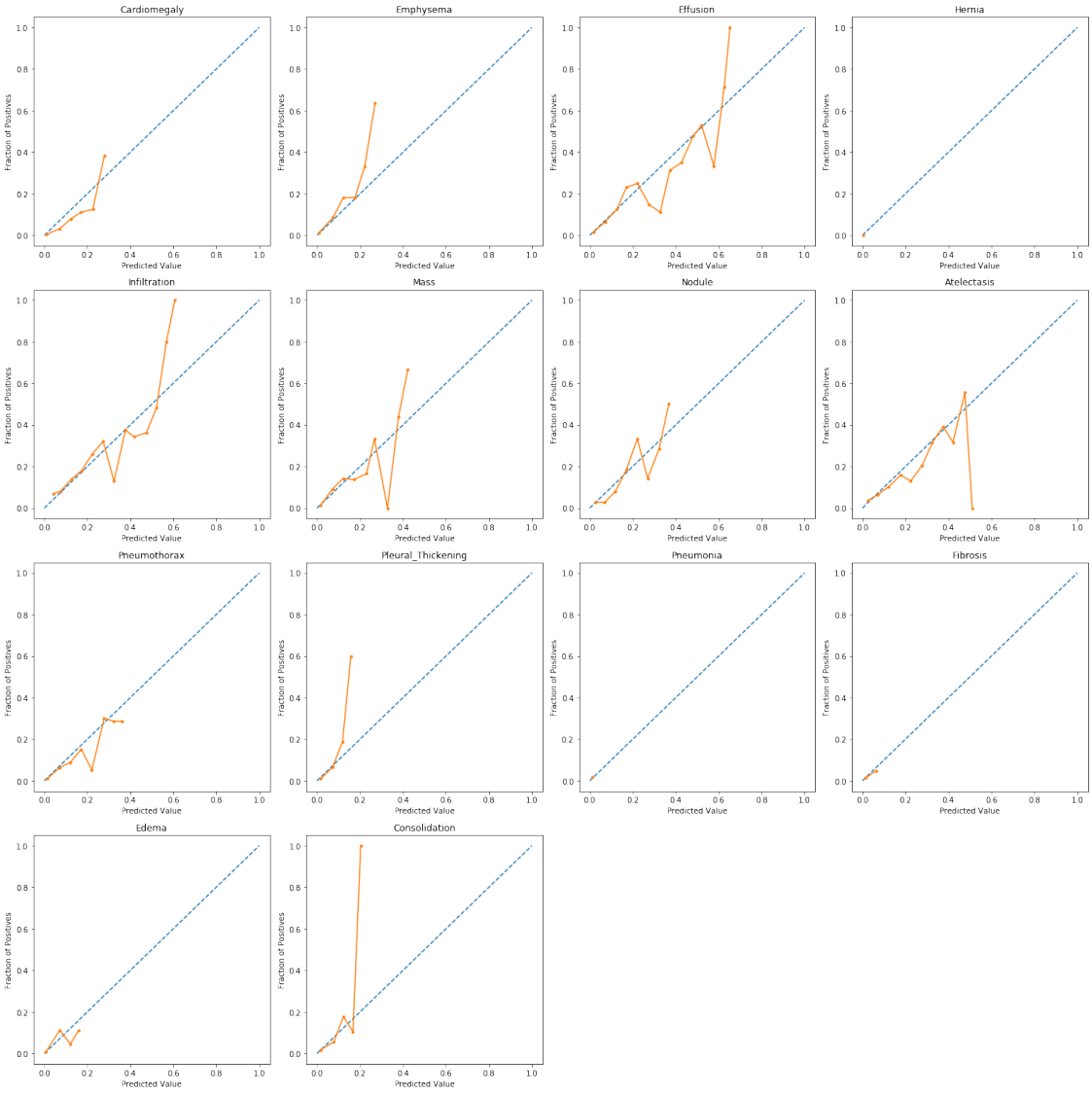
In [30]:
```
from sklearn.calibration import calibration_curve
def plot_calibration_curve(y, pred):
    plt.figure(figsize=(20, 20))
    for i in range(len(class_labels)):
        plt.subplot(4, 4, i + 1)
        fraction_of_positives, mean_predicted_value = calibration_curve(y[:,i], pred[
        plt.plot([0, 1], [0, 1], linestyle='--')
        plt.plot(mean_predicted_value, fraction_of_positives, marker='.')
        plt.xlabel("Predicted Value")
        plt.ylabel("Fraction of Positives")
        plt.title(class_labels[i])
    plt.tight_layout()
    plt.show()
```

In [31]:
```
plot_calibration_curve(y, pred)
```



In [32]:
```
from sklearn.linear_model import LogisticRegression as LR

    y_train = train_results[class_labels].values
    pred_train = train_results[pred_labels].values
    pred_calibrated = np.zeros_like(pred)

    for i in range(len(class_labels)):
        lr = LR(solver='liblinear', max_iter=10000)
        lr.fit(pred_train[:, i].reshape(-1, 1), y_train[:, i])
        pred_calibrated[:, i] = lr.predict_proba(pred[:, i].reshape(-1, 1))[:,1]
```

In [33]:
```
plot_calibration_curve(y[:,], pred_calibrated)
```



**********END**********

In [ ]: Project By: Aaditya Balakrishnan