In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import f1_score, roc_auc_score
```

In [2]:
```
model_df = pd.read_csv('model_preds2.csv')
gt_df = pd.read_csv('ground_truths.csv')
import pickle
with open('im_names.pkl', 'rb') as f:
    im_names = pickle.load(f)

# Actual per patient data
expert_df = pd.read_csv('bc4.csv')
truth_df = pd.read_csv('groundtruth.csv')
```

In [3]:
```
class_names = ['im_index', 'No Finding', 'Enlarged Cardiomediastinum', 'Cardiomegaly', 'Lung Opac
               'Lung Lesion', 'Edema', 'Consolidation', 'Pneumonia', 'Atelectasis', 'Pneumothorax
               'Pleural Effusion', 'Pleural Other', 'Fracture', 'Support Devices']
disease_names = ['No Finding', 'Enlarged Cardiomediastinum', 'Cardiomegaly', 'Lung Opacity',
               'Lung Lesion', 'Edema', 'Consolidation', 'Pneumonia', 'Atelectasis', 'Pneumothorax
               'Pleural Effusion', 'Pleural Other', 'Fracture', 'Support Devices']
model_df.columns = class_names
gt_df.columns = class_names
display(model_df)
```

	im_index	No Finding	Enlarged Cardiomediastinum	Cardiomegaly	Lung Opacity	Lung Lesion	Edema	Consolidation	P
0	0	0.019928	0.048625	0.181839	0.588069	0.013173	0.478689	0.068666	
1	1	0.193974	0.034552	0.006043	0.320518	0.060415	0.082389	0.031442	
2	2	0.012898	0.064089	0.329957	0.609478	0.031491	0.412228	0.076130	
3	3	0.261185	0.053228	0.263827	0.204052	0.022981	0.039993	0.020039	
4	4	0.080713	0.058124	0.338124	0.389094	0.029823	0.208432	0.079045	
...	...	...	...	...	...	...	...	...	
663	663	0.015705	0.023166	0.402352	0.611946	0.014530	0.709436	0.063083	
664	664	0.385992	0.029533	0.009237	0.114612	0.026862	0.008248	0.010091	
665	665	0.096112	0.041403	0.034940	0.371981	0.036817	0.083733	0.029535	
666	666	0.002200	0.014240	0.794432	0.557018	0.003705	0.875447	0.079249	
667	667	0.017503	0.027788	0.016902	0.589744	0.038803	0.073476	0.098707	

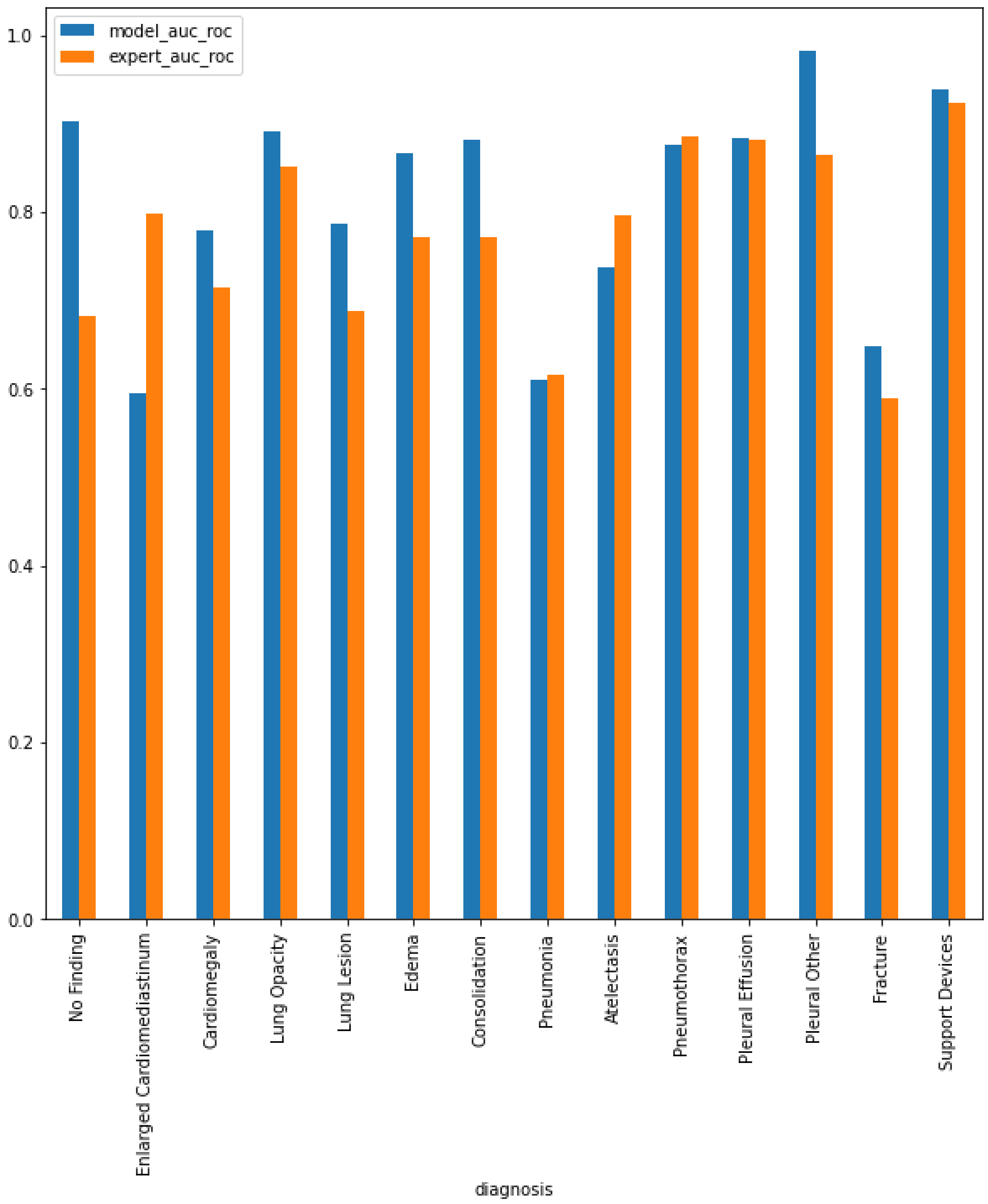
668 rows × 15 columns

In [4]:
```
def add_df_disease_metrics (true_df, pred_df, metric_df = None, col_name = None, metric_func=None
    if metric_func == None: # gets list_true, list_pred
        metric_func = roc_auc_score
    if type(metric_df) == type(None):
        metric_df = pd.DataFrame()
        metric_df['diagnosis'] = disease_names
    metric_list = []
    for diagnosis in disease_names:
        metric = metric_func(list(true_df[diagnosis]),list(pred_df[diagnosis]))
        metric_list.append(metric)
    metric_df[col_name] = metric_list
    return metric_df
```

In [5]:
```
patient_ids = []
fixed = []
for lsname in im_names:
    fixed.extend(lsname)
print(fixed[0])
print(len(fixed))
for fileName in fixed:
    #print(fileName.split('/')[2])
    pid = fileName.split('/')[2]
    patient_ids.append(pid)

# Patient sum probability diagnosis
model_df['pid']=patient_ids
patient_model_df = model_df.groupby('pid').max().drop(['im_index'], axis=1)

gt_df['pid']=patient_ids
patient_gt_df = gt_df.groupby('pid').max().drop(['im_index'], axis=1)


metrics_df = add_df_disease_metrics(true_df=patient_gt_df, pred_df=patient_model_df, col_name =
add_df_disease_metrics(true_df=truth_df, pred_df=expert_df, metric_df=metrics_df, col_name = 'ex

#sns.barplot(data=metrics_df, x='diagnosis', y='model_auc_roc')
metrics_df.plot.bar(x='diagnosis', y=['model_auc_roc', 'expert_auc_roc'], rot=90, figsize=(10,10
```

```
D:\WORK\EthicalRAI\project_data\CheXpert-v1.0-small/test/patient64741/study1/view1_frontal.jpg
668
```

Out[5]:
```
<AxesSubplot:xlabel='diagnosis'>
```



In [6]:
```
model_df['pid']=patient_ids
model_df_per_patient = model_df.groupby('pid').sum().drop(['im_index'], axis=1)
#display(model_df)
```

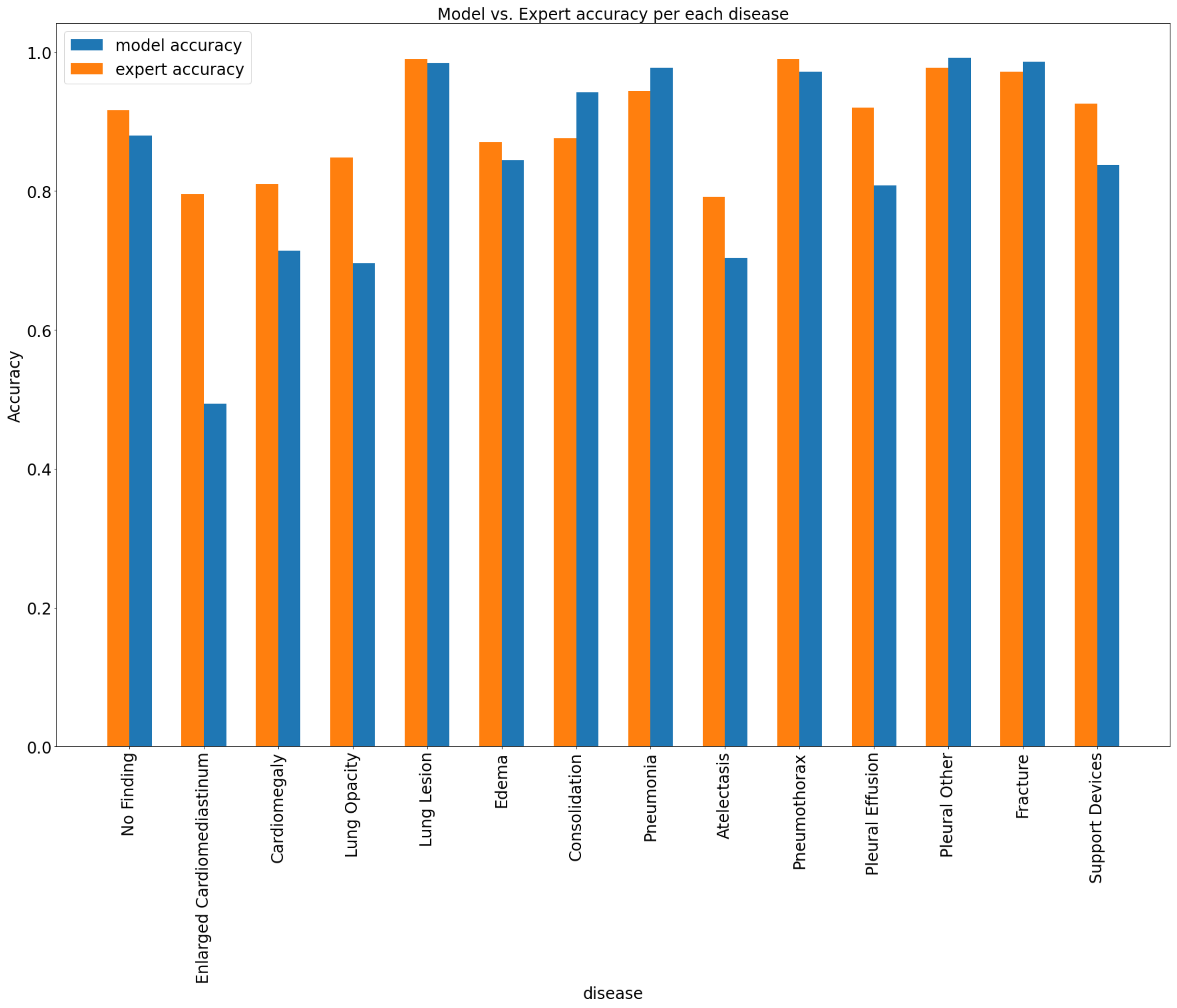
In [7]:
```
def check_accuracy(gt, df_pred):
    accuracy_list = []
    for col in disease_names:
        accuracy_per_desease = []
        for patient_gt, patient_pred in zip(gt[col], df_pred[col]):
            #print(str(df_pred[patient][col]))
            #print(str(gt[patient][col]))
            if (patient_pred>0.5 and patient_gt==1) or (patient_pred<=0.5 and patient_gt==0):
                accuracy_per_desease.append(1)
            else:
                accuracy_per_desease.append(0)
        accuracy_list.append(accuracy_per_desease)
    return accuracy_list
```

```
In [8]: def print_accuracy_per_desease(accu_list):
            final_accu = []
            for desease_index in range(len(disease_names)):
                accu = (sum(accu_list[desease_index]))/len(accu_list[desease_index])
                final_accu.append(accu)
                print("accuracy per disease " + str(disease_names[desease_index]) + " is: " + str(accu))
            return final_accu
```

```
In [9]: expert_accuracy_list = check_accuracy(truth_df, expert_df)
        print ("expert accuracies: ")
        expert_accuracy_final = print_accuracy_per_desease(expert_accuracy_list)

        model_accuracy_list = check_accuracy(truth_df, model_df_per_patient)
        print ("model accuracies: ")
        model_accuracy_final = print_accuracy_per_desease(model_accuracy_list)
```

```
expert accuracies:
accuracy per disease No Finding is: 0.916
accuracy per disease Enlarged Cardiomediastinum is: 0.796
accuracy per disease Cardiomegaly is: 0.81
accuracy per disease Lung Opacity is: 0.848
accuracy per disease Lung Lesion is: 0.99
accuracy per disease Edema is: 0.87
accuracy per disease Consolidation is: 0.876
accuracy per disease Pneumonia is: 0.944
accuracy per disease Atelectasis is: 0.792
accuracy per disease Pneumothorax is: 0.99
accuracy per disease Pleural Effusion is: 0.92
accuracy per disease Pleural Other is: 0.978
accuracy per disease Fracture is: 0.972
accuracy per disease Support Devices is: 0.926
model accuracies:
accuracy per disease No Finding is: 0.88
accuracy per disease Enlarged Cardiomediastinum is: 0.494
accuracy per disease Cardiomegaly is: 0.714
accuracy per disease Lung Opacity is: 0.696
accuracy per disease Lung Lesion is: 0.984
accuracy per disease Edema is: 0.844
accuracy per disease Consolidation is: 0.942
accuracy per disease Pneumonia is: 0.978
accuracy per disease Atelectasis is: 0.704
accuracy per disease Pneumothorax is: 0.972
accuracy per disease Pleural Effusion is: 0.808
accuracy per disease Pleural Other is: 0.992
accuracy per disease Fracture is: 0.986
accuracy per disease Support Devices is: 0.838
```

```
In [10]: X = disease_names
         ycont = expert_accuracy_final
         ztest = model_accuracy_final

         X_axis = np.arange(len(X))
         plt.figure(figsize=(30, 20))
         #plt.bar(X_axis, ycont, 0.4, label = 'model accuracy')
         #plt.bar(X_axis, ztest, 0.4, label = 'expert accuracy')

         plt.bar(X_axis + 0.15, ztest, 0.3, label = 'model accuracy')
         plt.bar(X_axis - 0.15, ycont, 0.3, label = 'expert accuracy')

         plt.tick_params(axis='x',  labelsize=24)
         plt.tick_params(axis='y', labelsize=24)
plt.xticks(X_axis, X,rotation='vertical')
plt.xlabel("disease",fontsize=24)
plt.ylabel("Accuracy",fontsize=24)
plt.title("Model vs. Expert accuracy per each disease",fontsize=24)
plt.legend(fontsize=24)
plt.show()
```

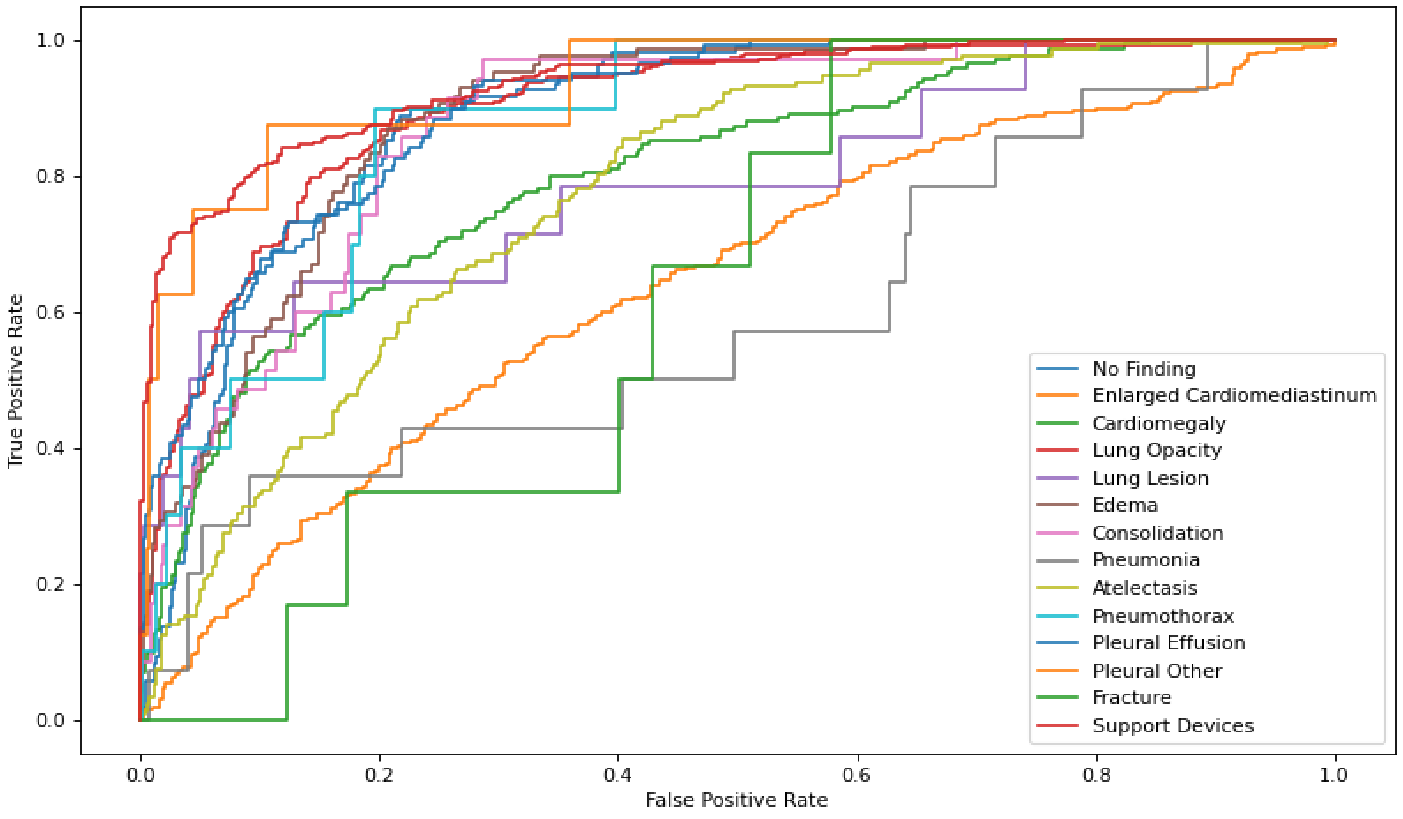


In [11]:
```
from sklearn.metrics import roc_curve

def plot_roc_curve(true_y, y_prob):
    """
    plots the roc curve based of the probabilities
    """

    fpr, tpr, thresholds = roc_curve(true_y, y_prob)
    plt.plot(fpr, tpr)
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    print(len(thresholds))



plt.figure(figsize=(12, 7), dpi=80)
```

```
for dn in disease_names:
    plot_roc_curve(gt_df[dn], model_df[dn])

plt.legend(disease_names)
```

```
122
319
206
172
26
111
58
29
214
22
126
15
15
142
```

Out[11]: `<matplotlib.legend.Legend at 0x2886e174580>`



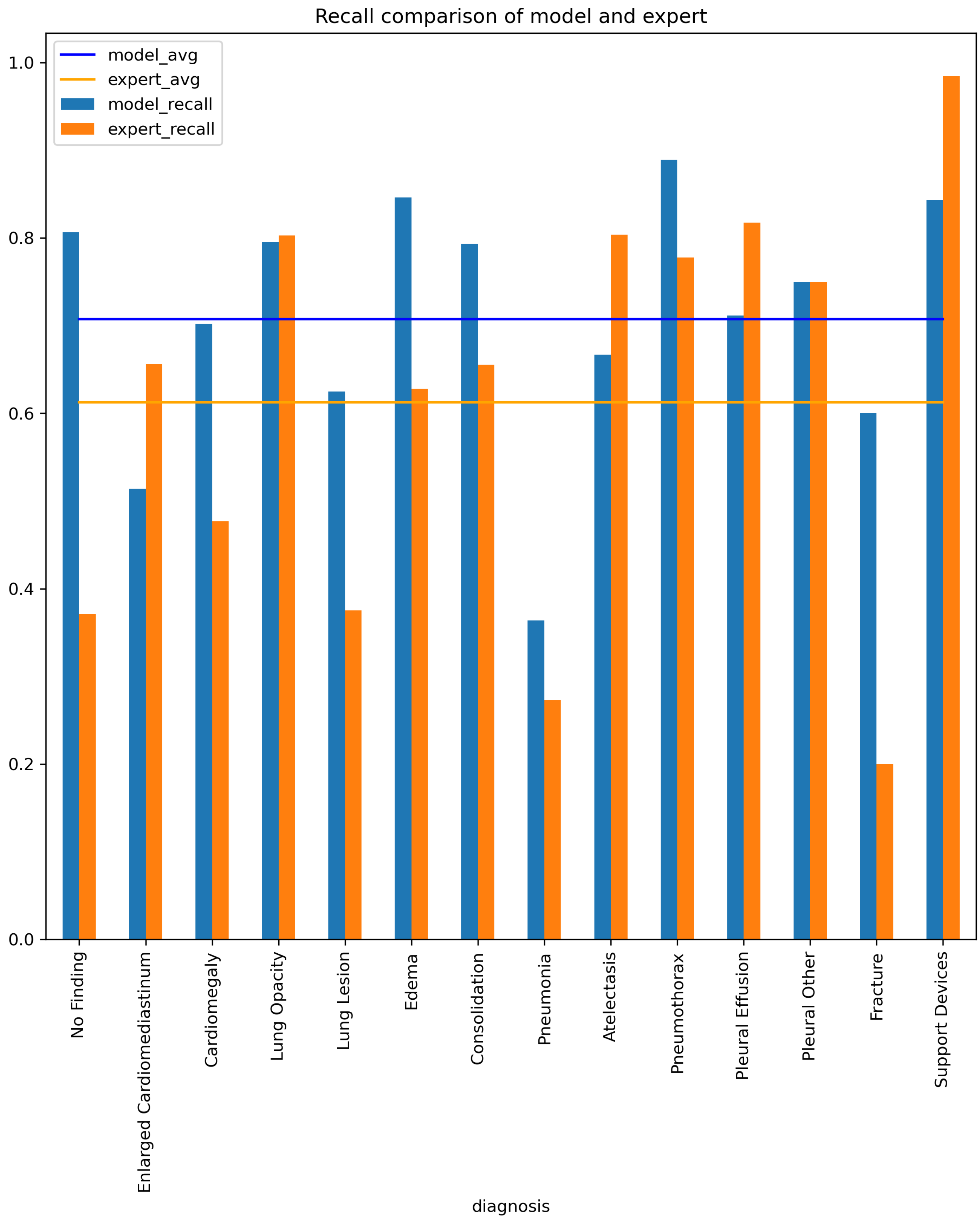
In [12]:
```
# find threshold per disease
import numpy as np
def find_apply_thresh(true_df, pred_df, data_df):
    out_df = pd.DataFrame()
    for diagnosis in disease_names:
        x,y,thres = roc_curve(list(true_df[diagnosis]),list(pred_df[diagnosis]))
        source = np.array((0,1))
        best_thres = thres[np.argmin([ np.linalg.norm( source - p) for p in zip(x,y)])] * 1.05
        print(f'For {diagnosis} chosen threshold {best_thres}')
        out_df[diagnosis] = data_df[diagnosis].apply(lambda x: 1 if x>=best_thres else 0)

    return out_df

patient_model_df_threshed = find_apply_thresh(true_df=patient_gt_df, pred_df=patient_model_df, d
For No Finding chosen threshold 0.27822637499999997
For Enlarged Cardiomediastinum chosen threshold 0.0402173163
For Cardiomegaly chosen threshold 0.0885403785
For Lung Opacity chosen threshold 0.4120154325
For Lung Lesion chosen threshold 0.0497333739
For Edema chosen threshold 0.20630536500000002
For Consolidation chosen threshold 0.083833449
For Pneumonia chosen threshold 0.0442058736
For Atelectasis chosen threshold 0.1780455285
For Pneumothorax chosen threshold 0.0761112345
For Pleural Effusion chosen threshold 0.47127990000000003
For Pleural Other chosen threshold 0.041214547500000004
For Fracture chosen threshold 0.0571100376
For Support Devices chosen threshold 0.4136034
```

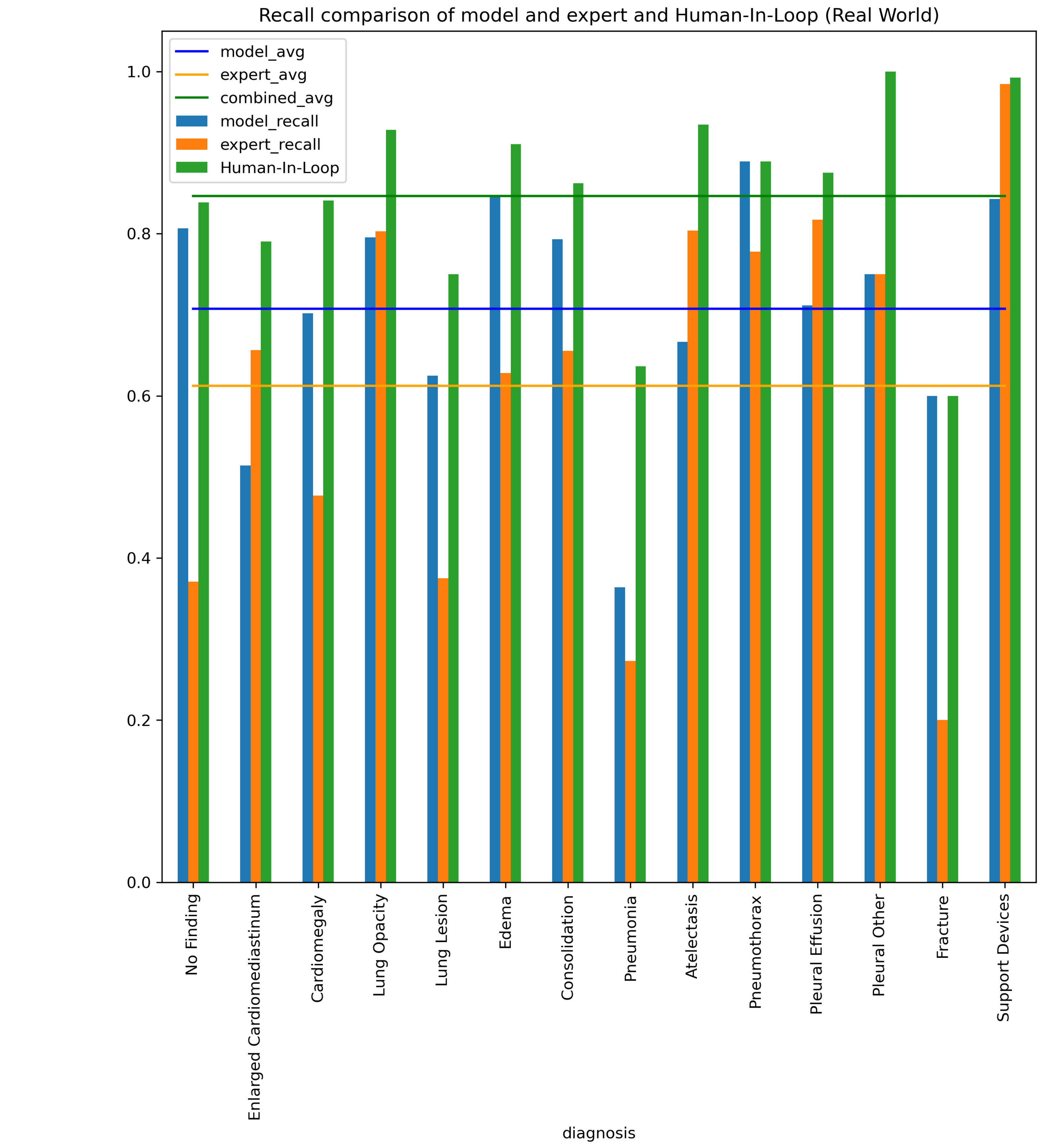
In [13]:
```
# Create combination of OR

combined_df = patient_model_df_threshed.reset_index(drop=True).add(expert_df[disease_names], fil
combined_df2 = patient_model_df_threshed.reset_index(drop=True).add(expert_df[disease_names], fi
```

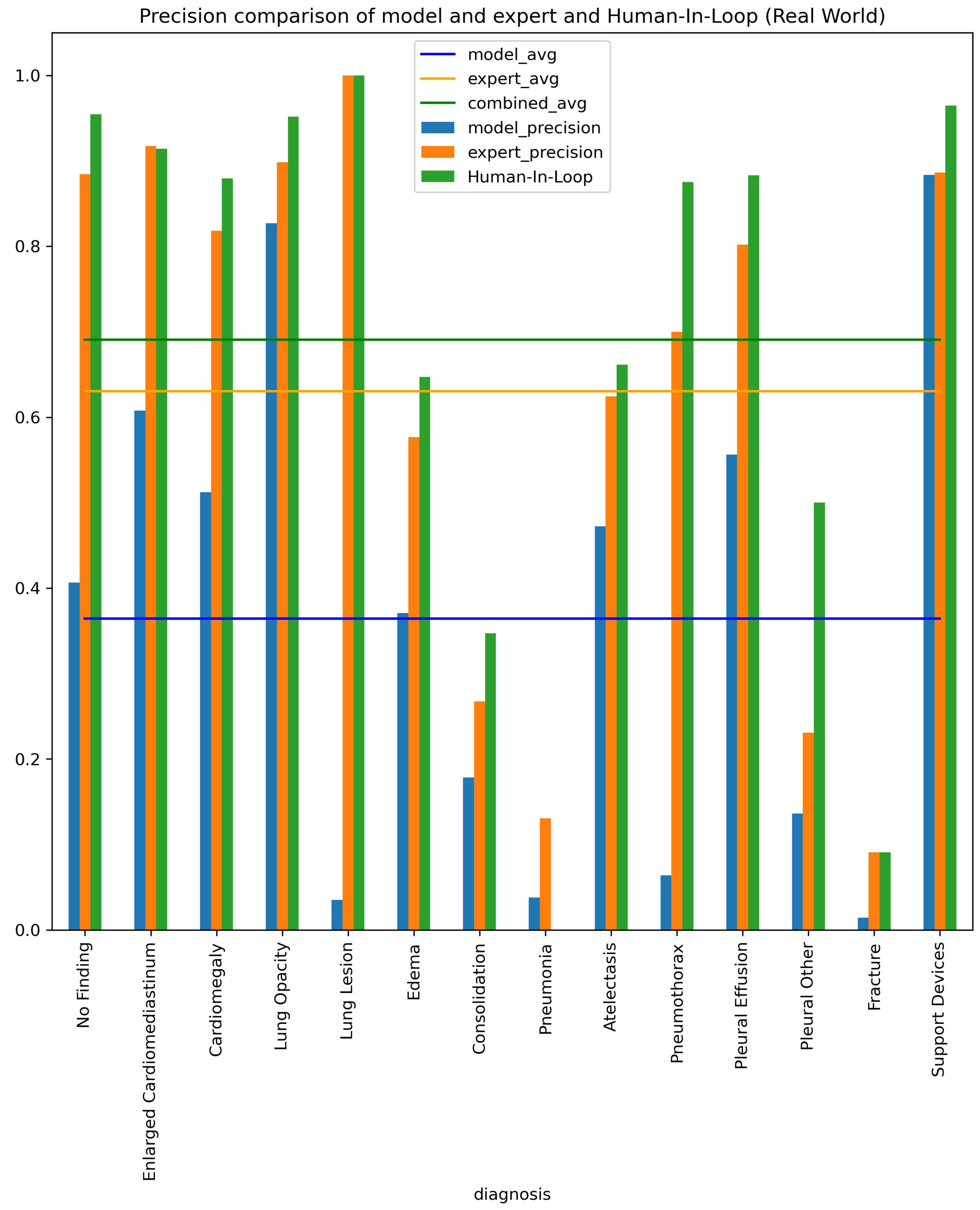
In [14]:
```
from sklearn.metrics import precision_score, f1_score, accuracy_score, recall_score, roc_auc_scor

def create_comparison(score_func,title_model,title_human, titleComb='Human-In-Loop', show_combin

    metrics_df = add_df_disease_metrics(true_df=patient_gt_df, pred_df=patient_model_df_threshed

    add_df_disease_metrics(true_df=truth_df, pred_df=expert_df, metric_df=metrics_df, col_name =

    add_df_disease_metrics(true_df=truth_df, pred_df=combine_df, metric_df=metrics_df, col_name

    if show_combin:
        metrics_df.plot.bar(x='diagnosis', y=[title_model, title_human, titleComb], rot=90,figsi
    else:
        metrics_df.plot.bar(x='diagnosis', y=[title_model, title_human], rot=90,figsize=(10,10))

    if give_auc_roc:
        model_auc_roc = 'model_AUC_ROC'
        add_df_disease_metrics(true_df=patient_gt_df, pred_df=patient_model_df, col_name = model
        expert_auc_roc = 'expert_AUC_ROC'
        add_df_disease_metrics(true_df=truth_df, pred_df=expert_df, col_name = expert_auc_roc, m
        if True:
            print(f'{expert_auc_roc}_avg: {np.mean(list(metrics_df[expert_auc_roc]))*100:.2f}%')
            print(f'{model_auc_roc}_avg: {np.mean(list(metrics_df[model_auc_roc]))*100:.2f}%')


    model_avg = np.mean(list(metrics_df[title_model]))
    expert_avg = np.mean(list(metrics_df[title_human]))
    combined_avg = np.mean(list(metrics_df[titleComb]))
    if print_avg:
        print(f'Only human average: {expert_avg*100:.2f}%')
        print(f'Only model average: {model_avg*100:.2f}%')

    plt.plot(np.arange(0,14), [model_avg]*len(np.arange(0,14)), label='model_avg', c='blue')
    plt.plot(np.arange(0,14), [expert_avg]*len(np.arange(0,14)), label='expert_avg', c='orange')
    if show_combin:
        plt.plot(np.arange(0,14), [combined_avg]*len(np.arange(0,14)), label='combined_avg', c='
    plt.legend()

plt.rcParams["figure.dpi"] = 300
plt.rcParams["figure.figsize"] = [12, 12]
```

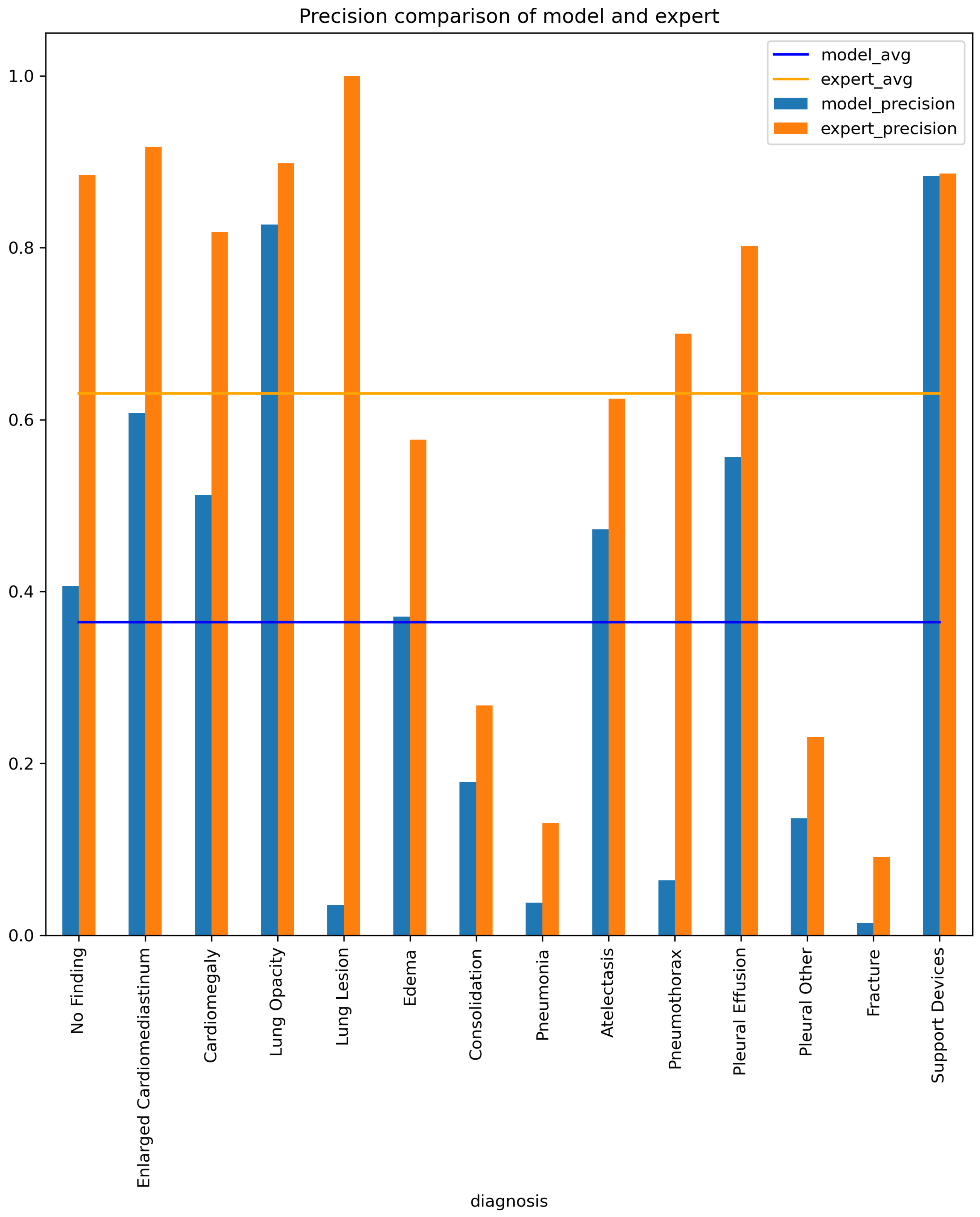
```
create_comparison(recall_score,'model_recall','expert_recall', show_combin=False, print_avg=True
plt.title('Recall comparison of model and expert')
plt.savefig('appendix_1.png', bbox_inches='tight')

create_comparison(recall_score,'model_recall','expert_recall')
plt.title('Recall comparison of model and expert and Human-In-Loop (Real World)')
plt.savefig('appendix_2.png', bbox_inches='tight')

create_comparison(precision_score,'model_precision','expert_precision', combine_df=combined_df2)
plt.title('Precision comparison of model and expert and Human-In-Loop (Real World)')
plt.savefig('appendix_3.png', bbox_inches='tight')

create_comparison(precision_score,'model_precision','expert_precision', show_combin=False)
plt.title('Precision comparison of model and expert')
plt.savefig('appendix_4.png', bbox_inches='tight')

# create_comparison(f1_score,'model_f1','expert_f1')
# plt.title('F1 Score comparison of model and expert')
#
# create_comparison(accuracy_score,'model_accuracy','expert_accuracy')
# plt.title('Accuracy comparison of model and expert')
#
# create_comparison(accuracy_score,'model_accuracy','expert_accuracy')
# plt.title('Accuracy comparison of model and expert')


# Stage 0:
    # Model is better in average recall - we choose to convert only to model.

# Stage 1:
    # People go to ER for enlarged cardio - even though AI said human does not have it

# Audit:
    # We look at a per-disease comparison -
    # Seeing that the AI performs significantly worse than human on some diseases

# Fix:
    # Solution - having a human in the loop to double check the AI results - achieve better than
    # Human and AI collaboration yields the best results!
```

```
expert_AUC_ROC_avg: 77.39%
model_AUC_ROC_avg: 81.30%
Only human average: 61.23%
Only model average: 70.75%
```


Recall comparison of model and expert
model_avg
expert_avg
model_recall
expert_recall
1.0
0.8
0.6
0.4
0.2
0.0
No Finding
Enlarged Cardiomediastinum
Cardiomegaly
Lung Opacity
Lung Lesion
Edema
Consolidation
Pneumonia
Atelectasis
Pneumothorax
Pleural Effusion
Pleural Other
Fracture
Support Devices
diagnosis


Recall comparison of model and expert and Human-In-Loop (Real World)
model_avg
expert_avg
combined_avg
model_recall
expert_recall
Human-In-Loop
0.0
0.2
0.4
0.6
0.8
1.0
No Finding
Enlarged Cardiomediastinum
Cardiomegaly
Lung Opacity
Lung Lesion
Edema
Consolidation
Pneumonia
Atelectasis
Pneumothorax
Pleural Effusion
Pleural Other
Fracture
Support Devices
diagnosis


Precision comparison of model and expert and Human-In-Loop (Real World)
model_avg
expert_avg
combined_avg
model_precision
expert_precision
Human-In-Loop
1.0
0.8
0.6
0.4
0.2
0.0
No Finding
Enlarged Cardiomediastinum
Cardiomegaly
Lung Opacity
Lung Lesion
Edema
Consolidation
Pneumonia
Atelectasis
Pneumothorax
Pleural Effusion
Pleural Other
Fracture
Support Devices
diagnosis



Precision comparison of model and expert
model_avg
expert_avg
model_precision
expert_precision
0.0
0.2
0.4
0.6
0.8
1.0
No Finding
Enlarged Cardiomediastinum
Cardiomegaly
Lung Opacity
Lung Lesion
Edema
Consolidation
Pneumonia
Atelectasis
Pneumothorax
Pleural Effusion
Pleural Other
Fracture
Support Devices
diagnosis

```
In [ ]:
```