SIIM-FISSIIM-FISABIO-RSNA COVID-19 Detection

Identify and localize COVID-19 abnormalities on chest radiographs

PROJECT

Submitted in partial fulfillment of the requirement for the award of the degree of

MASTER'S IN DATA SCIENCE

In course

DTSC 870-M01

by

Edara SaiSrija: 1275300
Mohmmed Safwan Aslam Kazi : 1270795
Mehreet Singh Bajaj : 1274698
Maitry Jariwala: 1275288

New York Institute of Technology

DECLARATION

We hereby declare that the projects entitled "***SIIM-FISSIIM-FISABIO-RSNA COVID-19 Detection (Identify and localize COVID-19 abnormalities on chest radiographs)*"** submitted as part of the partial course requirements for the course ***PROJECT (DTSC 870-M01)***, for the award of the degree of Master's in data science at New York Institute of Technology has been carried out by us as a group. We declare that the project has not formed the basis for the award of any degree, associate ship, fellowship or any other similar titles elsewhere.

Further, we declare that we will not share, re-submit or publish the code, idea, framework and/or any publication that may arise out of this work for academic or profit purposes without obtaining the prior written consent of the Course Faculty Mentor and Course Instructor.

MEHREET SINGH BAJAJ (1274698)

MAITRY JARIWALA (1275288)

EDARA SAISRIJA (1275300)

SAFWAN MOHMMAD KAZI (1270795)

Place: New York Institute of Technology

CERTIFICATE

This is to certify that the projects entitled " ***SIIM-FISSIIM-FISABIO-RSNA COVID-19 Detection(Identify and localize COVID-19 abnormalities on chest radiographs)*** " is a record of bonafide work carried out as part of the course ***PROJECT (DTSC 870 M01)***, under my guidance by Mehreet Singh Bajaj (1274698), Maitry Jariwala (1275288), Edara Saisrija (1275300), Safwan Mohmmad Kazi (1270795) during the academic semester ***4th***, in partial fulfilment of the requirements for the award of the Degree of Masters in Data Science, at New York Institute of Technology during academic year 2021.

JERRY CHENG
Project guide
New York Institute of Technology

ACKNOWLEDGEMENT

We as a group wish to express our deep sense of gratitude to our guide JERRY CHENG, Assistant Professor, Computer and Data Science department, New York Institute of Technology for his guidance and useful suggestions, for encouraging us to work on this project, using the techniques taught in the class and implementing them for real case scenarios, without which it would've been impossible to complete this project, that too in time. We are really thankful to him for believing in us and our project idea and helping us in shaping it up even better than we thought we could.

We feel blessed to have each other as a group of friends with same thinking and different ideologies which helped us in shaping the project perfectly in a fun and learning way. It was due to the intellectual discussion that we had with each other all the time that always motivated us and inspired us to do something, that we collaborated together and come up with the willingness to do these projects.

MEHREET SINGH BAJAJ (1274698)
MAITRY JARIWALA (1275288)
EDARA SAISRIJA (1275300)
SAFWAN MOHMMAD KAZI (1270795)
Master's in data science

ABSTRACT

SIIM-FISSIIM-FISABIO-RSNA COVID-19 Detection(Identify and localize COVID-19 abnormalities on chest radiographs)

COVID-19 causes significant morbidity and mortality. Like other pneumonias, pulmonary infection with COVID-19 results in inflammation and fluid in the lungs.

COVID-19 looks very similar to other viral and bacterial pneumonias on chest radiographs, which makes it difficult to diagnose. This can be diagnosed via polymerase chain reaction to detect genetic material from the virus or chest radiograph.

INTRODUCTION

1.1 Basic Introduction

Five times more deadly than the flu, COVID-19 causes significant morbidity and mortality. Like other pneumonias, pulmonary infection with COVID-19 results in inflammation and fluid in the lungs. COVID-19 looks very similar to other viral and bacterial pneumonias on chest radiographs, which makes it difficult to diagnose. Your computer vision model to detect and localize COVID-19 would help doctors provide a quick and confident diagnosis. As a result, patients could get the right treatment before the most severe effects of the virus take hold.

Currently, COVID-19 can be diagnosed via polymerase chain reaction to detect genetic material from the virus or chest radiograph. However, it can take a few hours and sometimes days before the molecular test results are back. By contrast, chest radiographs can be obtained in minutes. While guidelines exist to help radiologists differentiate COVID-19 from other types of infection, their assessments vary. In addition, non-radiologists could be supported with better localization of the disease, such as with a visual bounding box.

As the leading healthcare organization in their field, the Society for Imaging Informatics in Medicine (SIIM)'s mission is to advance medical imaging informatics through education, research, and innovation. SIIM has partnered with the Foundation for the Promotion of Health and Biomedical Research of Valencia Region (FISABIO), Medical Imaging Databank of the Valencia Region (BIMCV) and the Radiological Society of North America (RSNA) for this competition.

1.2 Scope of Work

This project will help radiologists diagnose the millions of COVID-19 patients more confidently and quickly. This will also enable doctors to see the extent of the disease and help them make decisions regarding treatment. Depending upon severity, affected patients may need hospitalization, admission into an intensive care unit, or supportive therapies like mechanical ventilation. As a result of better diagnosis, more patients will quickly receive the best care for their condition, which could mitigate the most severe effects of the virus.

REQUIREMENT ANALYSIS

2.1 Functional Requirements

Providing the details of the project, all requirements and specifications will be made. Some of the functionalities we are going to use in our project are the basic principles of Data science along with Python. Clustering data grouping techniques from Machine Learning i.e. KMeans and Visualizations using ggplot2, Re, wandb, tqdm , pandas, numpy, Seaborn, matplotlib, pyplot, offsetbox, AnnotationBbox, OffsetImage , pydicom . Most of the important libraries of python have been implemented in our code in the best possible way to enhance the quality of our project.

2.2Dataset

Data used was originally published by the Medical Imaging Databank of the Valencia Region (BIMCV) in cooperation with The Foundation for the Promotion of Health and Biomedical Research of Valencia Region (FISABIO), and the Regional Ministry of Innovation, Universities, Science and Digital Society (Generalitat Valenciano), however the images were completely re-annotated using different annotation types.

- The train dataset comprises 6,334 chest scans in DICOM format, which were de-identified to protect patient privacy.
- All images were labeled by a panel of experienced radiologists for the presence of opacities as well as overall appearance.
- All images are stored in paths with the form **STUDY/SERIES/IMAGE**. THE **study ID** relates to the study –level predictions and the **image ID** is the ID used for image level predictions.

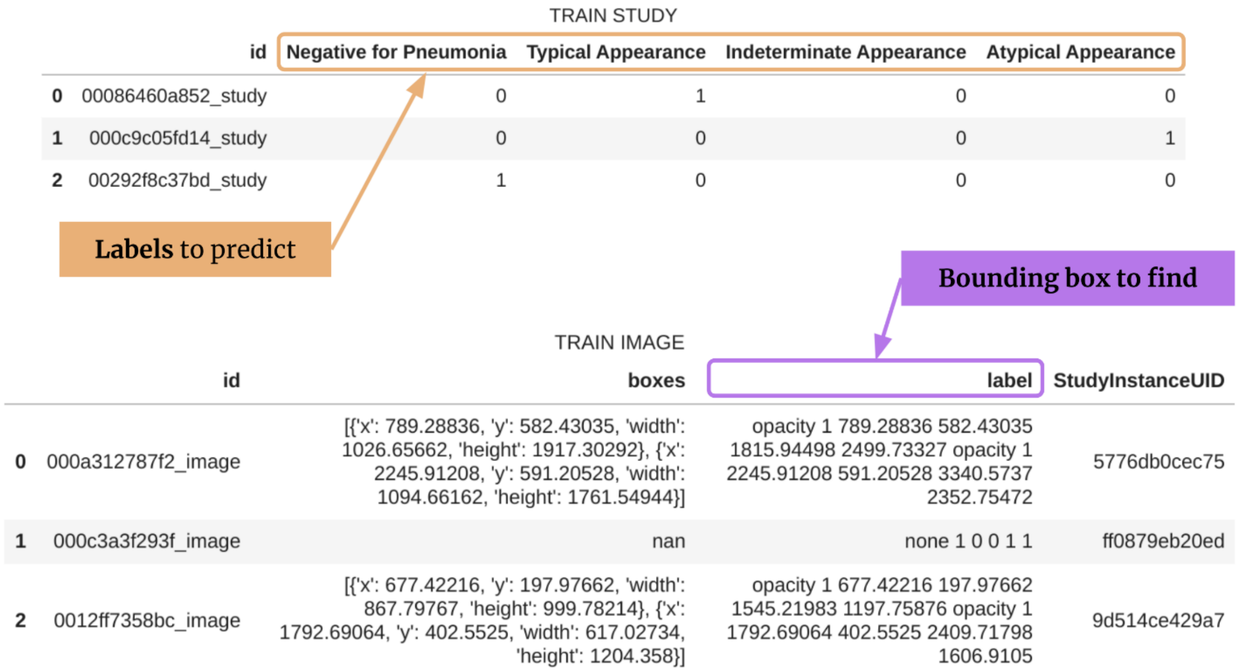
Dataset Link: *https://www.kaggle.com/c/siim-covid19-detection/data*

Metadata

Our data consists of images + .csv files, containing custom information for each radiography.

Metadata structure:

1. train_study_level.csv - contains one row for each study, including correct labels.
2. train_image_level.csv - containing one row for each image, including both correct labels and any bounding boxes in a dictionary format



TRAIN STUDY

	id	Negative for Pneumonia	Typical Appearance	Indeterminate Appearance	Atypical Appearance
0	00086460a852_study	0	1	0	0
1	000c9c05fd14_study	0	0	0	1
2	00292f8c37bd_study	1	0	0	0

TRAIN IMAGE

	id	boxes	label	StudyInstanceUID
0	000a312787f2_image	[{'x': 789.28836, 'y': 582.43035, 'width': 1026.65662, 'height': 1917.30292}, {'x': 2245.91208, 'y': 591.20528, 'width': 1094.66162, 'height': 1761.54944}]	opacity 1 789.28836 582.43035 1815.94498 2499.73327 opacity 1 2245.91208 591.20528 3340.5737 2352.75472	5776db0cec75
1	000c3a3f293f_image	nan	none 1 0 0 1 1	ff0879eb20ed
2	0012ff7358bc_image	[{'x': 677.42216, 'y': 197.97662, 'width': 867.79767, 'height': 999.78214}, {'x': 1792.69064, 'y': 402.5525, 'width': 617.02734, 'height': 1204.358}]	opacity 1 677.42216 197.97662 1545.21983 1197.75876 opacity 1 1792.69064 402.5525 2409.71798 1606.9105	9d514ce429a7

📌 **Important**: An image can have *multiple bounding boxes*.

In [5]:

```
# Save data to W&B Dashboard
save_dataset_artifact(run_name='save-train1',
                      artifact_name='train_study_level',
                      path="../input/siim-covid19-detection/train_study_level.csv")

save_dataset_artifact(run_name='save-train2',
                      artifact_name='train_image_level',
                      path="../input/siim-covid19-detection/train_image_level.csv")
```

In [6]:

```
# Read in metadata
train_study = pd.read_csv("../input/siim-covid19-detection/train_study_level.csv")
train_image = pd.read_csv("../input/siim-covid19-detection/train_image_level.csv")

print(color.BOLD + "Train Study Shape:" + color.END, train_study.shape, "\n" +
      color.BOLD + "Train Image Shape:" + color.END, train_image.shape, "\n" +
      "\n" +
      "Note: There are {} missing values in train_image.".\
                        format(train_image["boxes"].isna().sum()), "\n" +
      "This happens for labels = 'none' - no checkboxes.", 3*"\n")


# Head of our 2 training metadata
df1_styler = train_study.head(3).style.set_table_attributes("style='display:inline'").\
                                 set_caption('TRAIN STUDY')
df2_styler = train_image.head(3).style.set_table_attributes("style='display:inline'").\
                                 set_caption('TRAIN IMAGE')

display_html(df1_styler._repr_html_() + df2_styler._repr_html_(), raw=True)
```

```
Train Study Shape:
Train Image Shape:
```

TRAIN STUDY

	id	Negative for Pneumonia	Typical Appearance	Indeterminate Appearance	Atypical Appearance
0	00086460a852_study	0	1	0	0
1	000c9c05fd14_study	0	0	0	1
2	00292f8c37bd_study	1	0	0	0

TRAIN IMAGE

	id	boxes	label	StudyInstanceUID
0	000a312787f2_image	[{'x': 789.28836, 'y': 582.43035, 'width': 1026.65662, 'height': 1917.30292}, {'x': 2245.91208, 'y': 591.20528, 'width': 1094.66162, 'height': 1761.54944}]	opacity 1 789.28836 582.43035 1815.94498 2499.73327 opacity 1 2245.91208 591.20528 3340.5737 2352.75472	5776db0cec75
1	000c3a3f293f_image	nan	none 1 0 0 1 1	ff0879eb20ed
2	0012ff7358bc_image	[{'x': 677.42216, 'y': 197.97662, 'width': 867.79767, 'height': 999.78214}, {'x': 1792.69064, 'y': 402.5525, 'width': 617.02734, 'height': 1204.358}]	opacity 1 677.42216 197.97662 1545.21983 1197.75876 opacity 1 1792.69064 402.5525 2409.71798 1606.9105	9d514ce429a7

1.1 train_study analysis

🔎 **Findings**:

1. id: there are 6,054 unique ids - there are **no duplicates**
2. target: one of our targets is to predict is the radiography is negative_for_pneumonia, has typical appearance, has indeterminate appearance or it's just atypical.
3. an image can have **positive value for only 1 label**. For example, there aren't any images which are both negative_for_pneumonia and indeterminate appearance in the same time. It's only one or the other.
4. **class imbalance is present** - especially for Indeterminate Appearance and Atypical Appearance

Target labels distribution

Now let's see how the **labels we'll have to predict** are layed out.

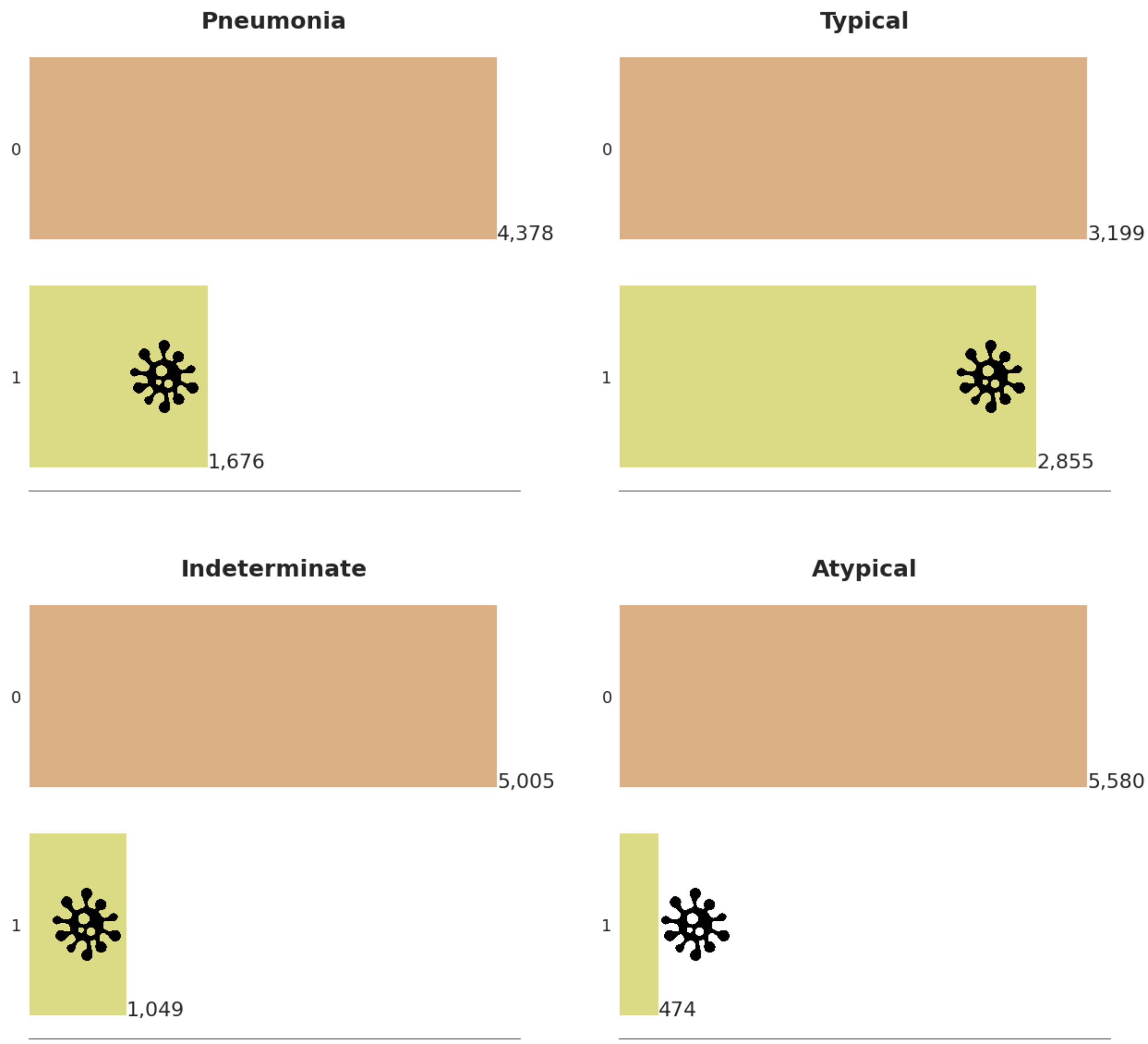
In [7]:

```
run = wandb.init(project='siim-covid19', name='metadata_eda', config=CONFIG,
anonymous="allow")
```

In [8]:

```
# Process id
train_study["study_id"] = train_study["id"].apply(lambda x: x.split("_")[0])

# Data for plots
pneumonia = train_study["Negative for Pneumonia"]
typical = train_study["Typical Appearance"]
indeterminate = train_study["Indeterminate Appearance"]
atypical = train_study["Atypical Appearance"]

# Plotting
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, figsize=(21,20))
axs = [ax1, ax2, ax3, ax4]
dfs = [pneumonia, typical, indeterminate, atypical]
titles = ["Pneumonia", "Typical", "Indeterminate", "Atypical"]
```

```
for ax, df, title in zip(axs, dfs, titles):
    sns.countplot(y=df, ax=ax, palette=my_colors[1:])
    ax.set_title(title, fontsize=25, weight='bold')
    show_values_on_bars(ax, h_v="h", space=0.4)
    ax.set_xticklabels([])
    ax.set_ylabel('')
    ax.set_xlabel('')

# Virus png
path='../input/siimfisabiorsna-covid-2021/PinClipart.com_virus-clip-art_742280.png'
offset_png(x=4378, y=1, path=path, ax=ax1, zoom=0.05, offset=-360, border=1)
offset_png(x=2855, y=1, path=path, ax=ax2, zoom=0.05, offset=-50, border=1)
offset_png(x=1049, y=1, path=path, ax=ax3, zoom=0.05, offset=-43, border=1)
offset_png(x=474, y=1, path=path, ax=ax4, zoom=0.05, offset=40, border=1)
```



In [9]:

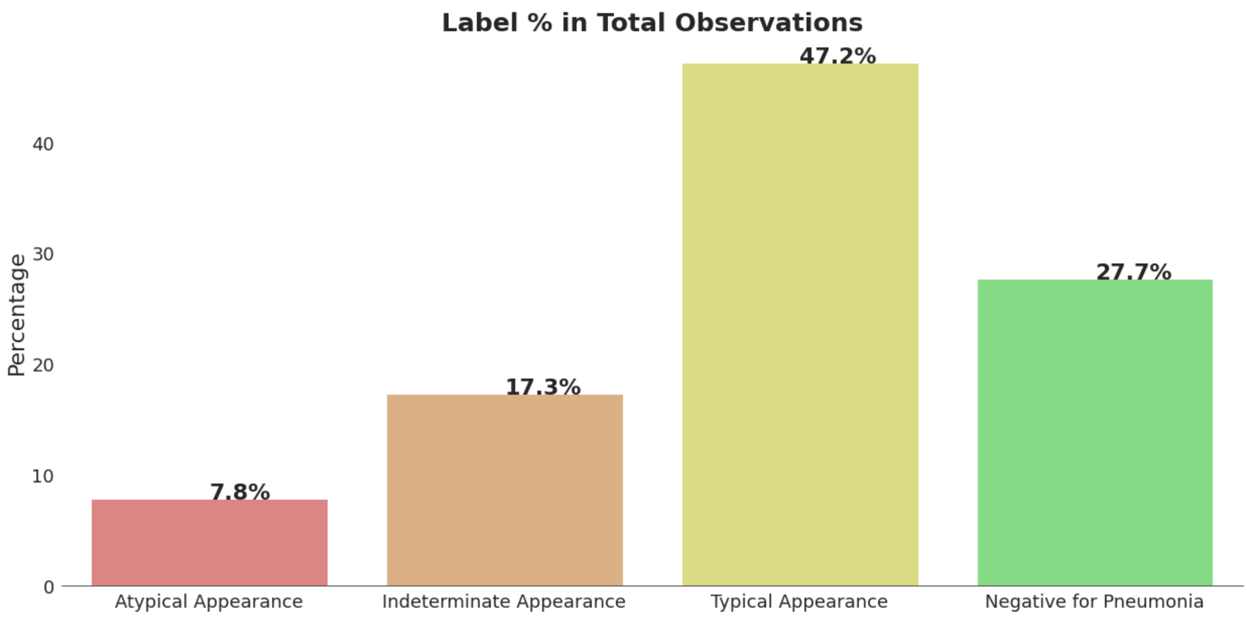
```
# Save plots into W&B Dashboard
for title, df in zip(titles, dfs):
    create_wandb_plot(x_data=[0, 1],
                      y_data=df.value_counts().values,
                      x_name="Flag", y_name="Freq", title=title,
                      log=title, plot="bar")
```

How many instances per label?

📌 **Note**: 50% of our images have typical appearance. The rest 50% is split in order into *negative for pneumonia* (28%), *indeterminate* (17%) and the rest *atypical*.

In [10]:

```
# Get data and transform frequencies to percentages
df = train_study.groupby(['Negative for Pneumonia', 'Typical Appearance',
    'Indeterminate Appearance', 'Atypical Appearance']).count().reset_index()

df["label"] = ['Atypical Appearance', 'Indeterminate Appearance',
        'Typical Appearance', 'Negative for Pneumonia']
df["perc"] = df["id"]/df["id"].sum()*100

# Plot
bar,ax = plt.subplots(figsize=(21,10))
ax = sns.barplot(x=df["label"], y=df["perc"],
         ci=None, palette=my_colors, orient='v')
ax.set_title("Label % in Total Observations", fontsize=25,
       weight = "bold")
ax.set_xlabel(" ")
ax.set_ylabel("Percentage")
for rect in ax.patches:
  ax.text (rect.get_x() + rect.get_width() / 2,rect.get_height(),
       "%.1f%%"% rect.get_height(), weight='bold')
```



In [11]:

```
create_wandb_plot(x_data=df["label"].values,
                  y_data=df["perc"].values,
                  x_name="Label", y_name="Percentage",
                  title="Label % in Total Observations",
                  log="perc", plot="bar")
```

In [12]:

```
wandb.finish()
```

1.2 train_image analysis

🔎 **Findings**:

1. StudyInstanceUID corresponds 1:1 to new_id created for train_study.
2. image_id is unique in the image_train data.
3. There are images with multiple bounding boxes!
4. There can be multiple images per study!

How many images have some sort of abnormality?

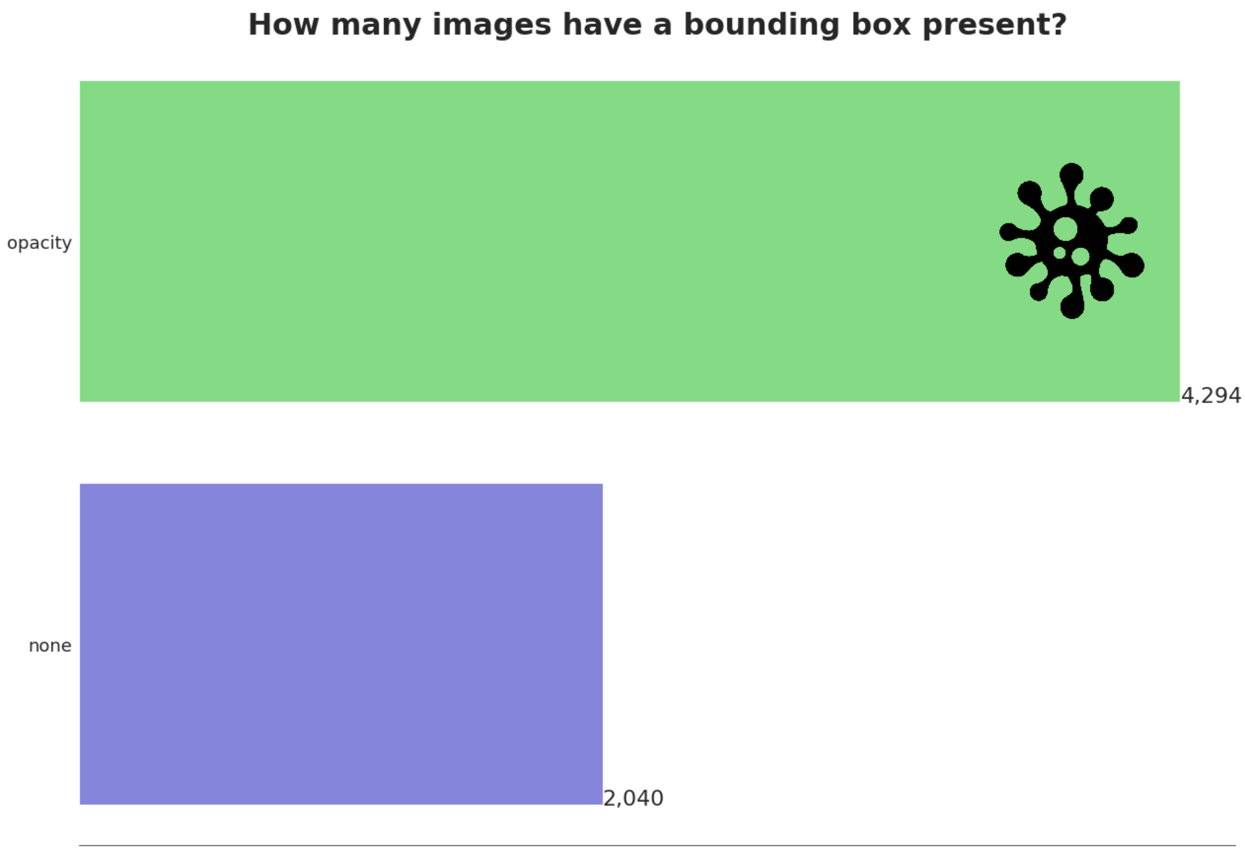
To have a bounding box we need to have something weird detected in the scan. If there is nothing weird ... then we don't need a bounding box!

In [13]:

```
run = wandb.init(project='siim-covid19', name='train_imgs_eda', config=CONFIG,
anonymous="allow")
```

In [14]:

```
# Process id
train_image["image_id"] = train_image["id"].apply(lambda x: x.split("_")[0])

# Data for plotting
df = train_image["label"].apply(lambda x: x.split(" ")[0]).\
                              value_counts().reset_index()

# Plot
plt.figure(figsize=(21, 15))
ax = sns.barplot(data=df, y="index", x="label", palette=my_colors[3:])
show_values_on_bars(ax, h_v="h", space=0.4)
plt.title("How many images have a bounding box present?",
       fontsize=30, weight='bold')
plt.xticks([])
plt.ylabel('')
plt.xlabel('');

# Virus png
path='../input/siimfisabiorsna-covid-2021/PinClipart.com_virus-clip-art_742280.png'
offset_png(x=4294, y=0, path=path, ax=ax, zoom=0.1, offset=-110, border=1)
```



In [15]:

```
create_wandb_plot(x_data=df["index"],
                  y_data=df["label"].values,
                  x_name="BBox", y_name="Freq",
                  title="Images with bbox",
                  log="bbox", plot="bar")
```

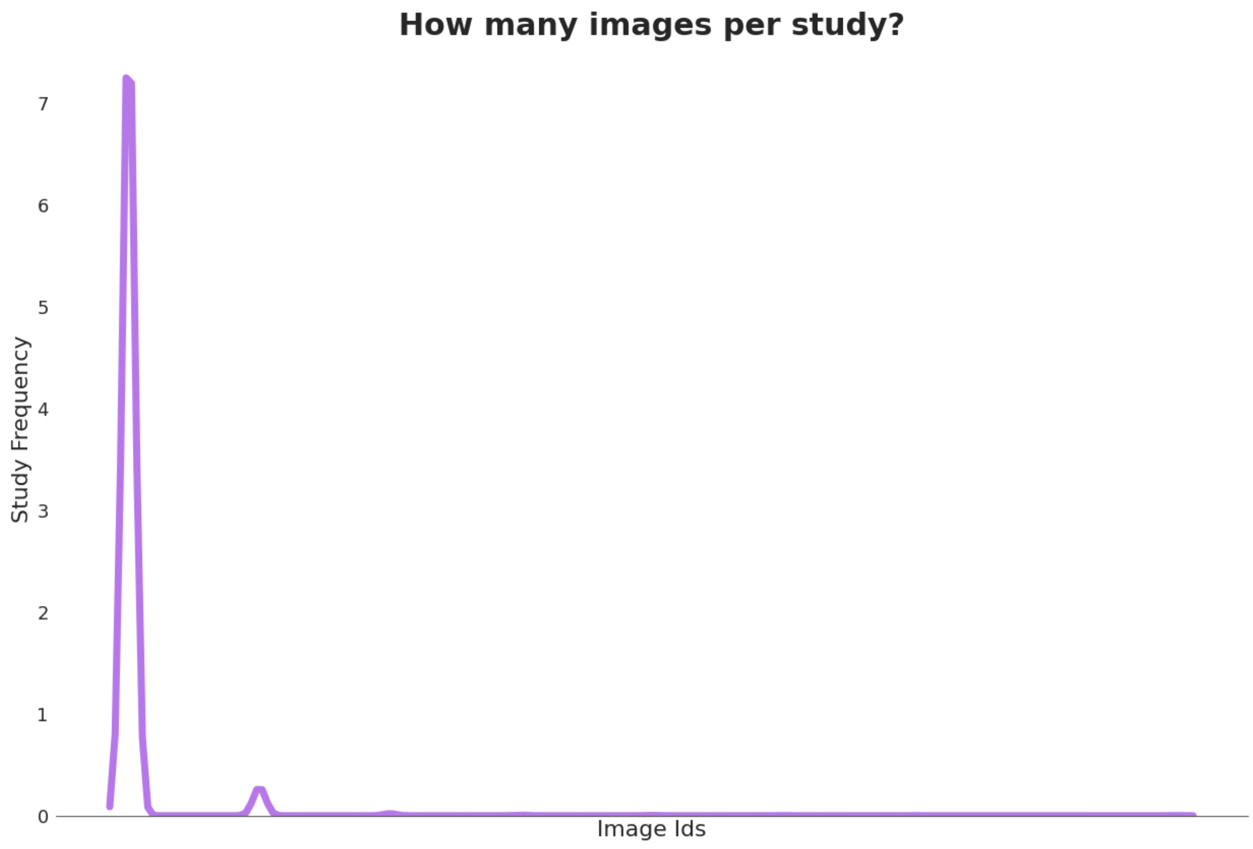
How many images per each study?

Majority of studies have only 1 images. That being said, we have ~230 studies that have multiple images available (up to 9).

In [16]:

```
# Crate df
df = train_image["StudyInstanceUID"].value_counts().reset_index().\
                        sort_values("StudyInstanceUID", ascending=False)
print(color.BOLD + "Max number of images available per study:" + color.END,
      df["StudyInstanceUID"].max(), "\n" +
      color.BOLD + "Min number of images available per study:" + color.END,
    df["StudyInstanceUID"].min(), 2*"\n")
# Plot
plt.figure(figsize=(21, 14))
sns.distplot(a=df["StudyInstanceUID"], color=my_colors[6],
             hist=False, kde_kws=dict(lw=7, ls="-"))
plt.title("How many images per study?",
          fontsize=30, weight='bold')
plt.xticks([])
plt.ylabel('Study Frequency')
plt.xlabel('Image Ids');
```

```
Max number of images available per study:
Min number of images available per study:
```



In [17]:

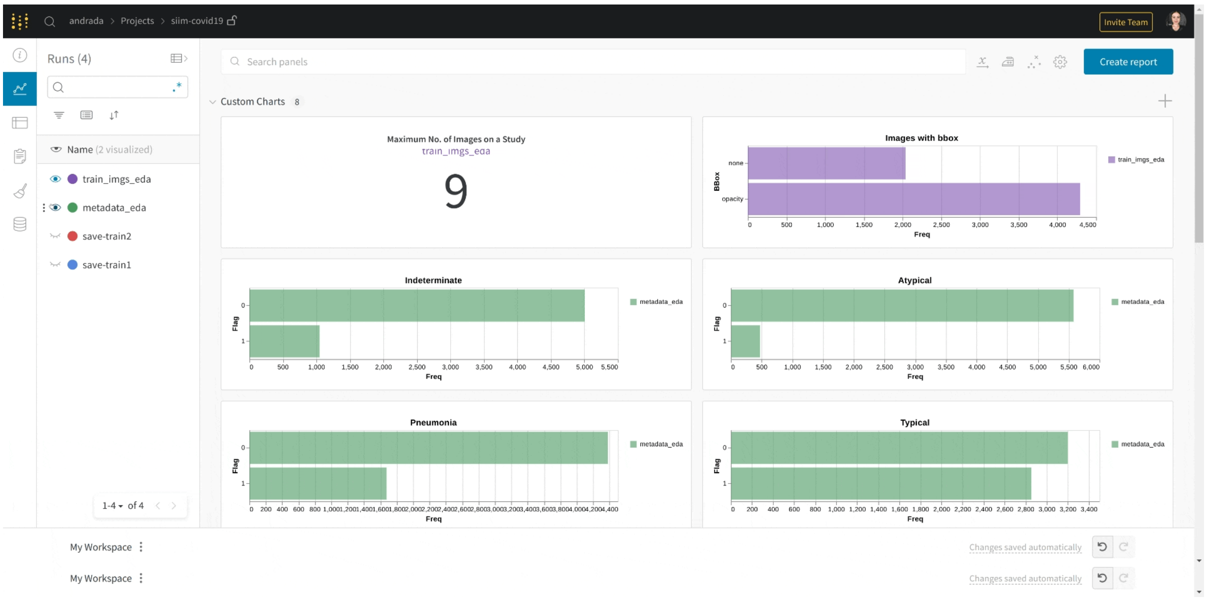
```
wandb.log({"max_images_on_study" : df["StudyInstanceUID"].max()})
```

```
create_wandb_hist(x_data=df["StudyInstanceUID"],
                  x_name="Image Freq", title="No. images per study",
                  log="hist")
```

In [18]:

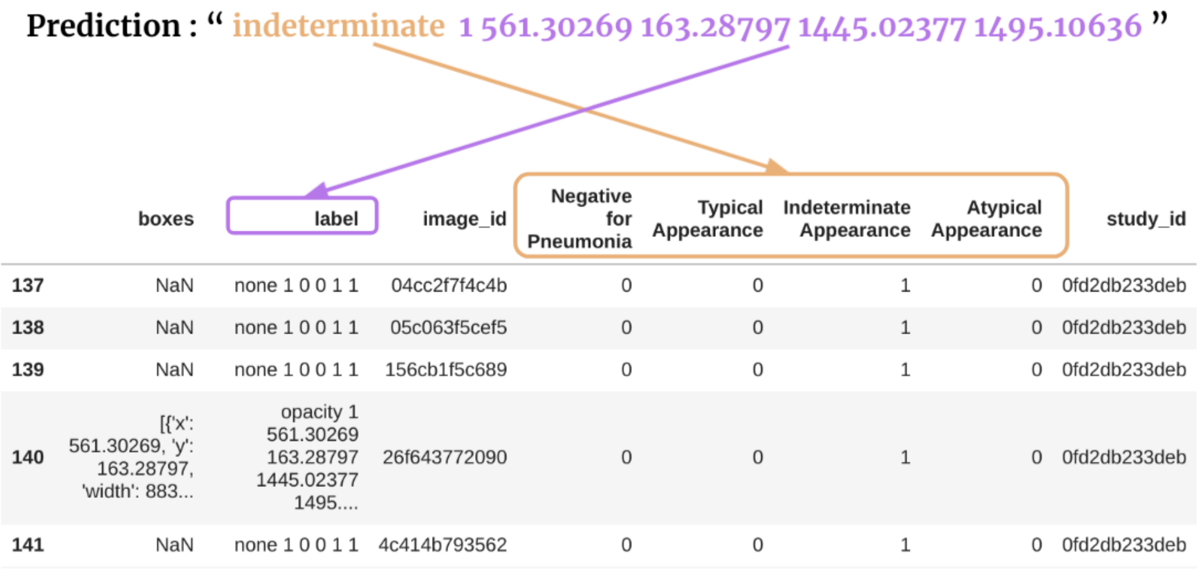
```
wandb.finish()
```

After EDA, my [W&B Dashboard] looks like this:



Create the full train dataset

This is also how the final_label will need to look before submission.



Prediction : " indeterminate 1 561.30269 163.28797 1445.02377 1495.10636 "

	boxes	label	image_id	Negative for Pneumonia	Typical Appearance	Indeterminate Appearance	Atypical Appearance	study_id
137	NaN	none 1 0 0 1 1	04cc2f7f4c4b	0	0	1	0	0fd2db233deb
138	NaN	none 1 0 0 1 1	05c063f5cef5	0	0	1	0	0fd2db233deb
139	NaN	none 1 0 0 1 1	156cb1f5c689	0	0	1	0	0fd2db233deb
140	[{'x': 561.30269, 'y': 163.28797, 'width': 883...	opacity 1 561.30269 163.28797 1445.02377 1495....	26f643772090	0	0	1	0	0fd2db233deb
141	NaN	none 1 0 0 1 1	4c414b793562	0	0	1	0	0fd2db233deb

In [19]:

```
# Merge all info together
train = pd.merge(train_image, train_study,
                 left_on="StudyInstanceUID", right_on="study_id")

train.drop(["id_x", "StudyInstanceUID", "id_y"], axis=1, inplace=True)
```

Let's also look at the study id 0fd2db233deb, which has most of the images available. 8 images have nothing unusual in them and 1 is labeled as Indeterminate Appearance.

In [20]: `train[train["study_id"] == "0fd2db233deb"]`

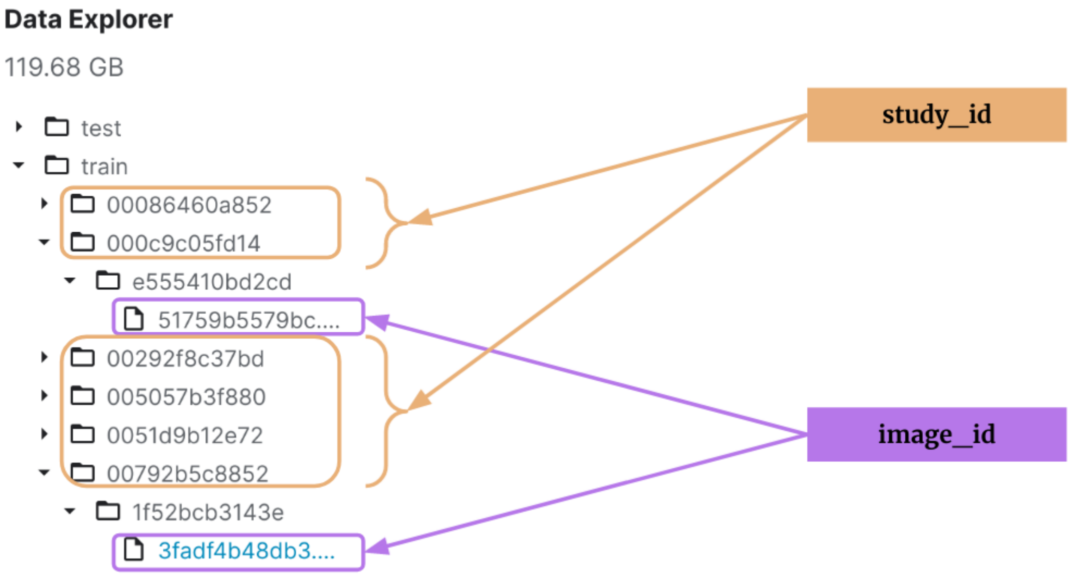
Out[20]:

	boxes	label	image_id	Negative for Pneumonia	Typical Appearance	Indeterminate Appearance	Atypical Appearance	study_id
137	NaN	none 1 0 0 1 1	04cc2f7f4c4b	0	0	1	0	0fd2db233deb
138	NaN	none 1 0 0 1 1	05c063f5cef5	0	0	1	0	0fd2db233deb
139	NaN	none 1 0 0 1 1	156cb1f5c689	0	0	1	0	0fd2db233deb
140	[{'x': 561.30269, 'y': 163.28797, 'width': 883...	opacity 1 561.30269 163.28797 1445.02377 1495....	26f643772090	0	0	1	0	0fd2db233deb
141	NaN	none 1 0 0 1 1	4c414b793562	0	0	1	0	0fd2db233deb
142	NaN	none 1 0 0 1 1	a5a364383f34	0	0	1	0	0fd2db233deb
143	NaN	none 1 0 0 1 1	b121806162c3	0	0	1	0	0fd2db233deb

144	NaN	none 1 0 0 1 1	bee62c601ae9	0	0	1	0	0fd2db233deb
145	NaN	none 1 0 0 1 1	c6e92e59a0ae	0	0	1	0	0fd2db233deb

2. Images

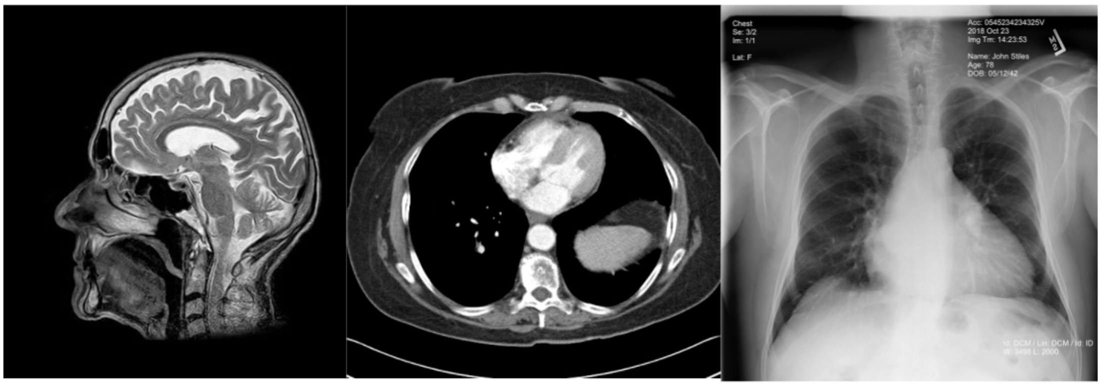
Good, now that we've explored the metadata and the target labels, we can start focusing on the good stuff - meaning the **CT scans**.



WHAT ARE CT SCANS?

"A **Computerized Tomography** scan (CT or CAT scan) uses computers and rotating X-ray machines to **create cross-sectional images** of the body. These images provide **more detailed** information than normal X-ray images. They can show the **soft tissues**, **blood vessels**, and **bones** in various parts of the body."

CT Scan Examples



In [21]:

```
run = wandb.init(project='siim-covid19', name='image_explore', config=CONFIG,
anonymous="allow")
```

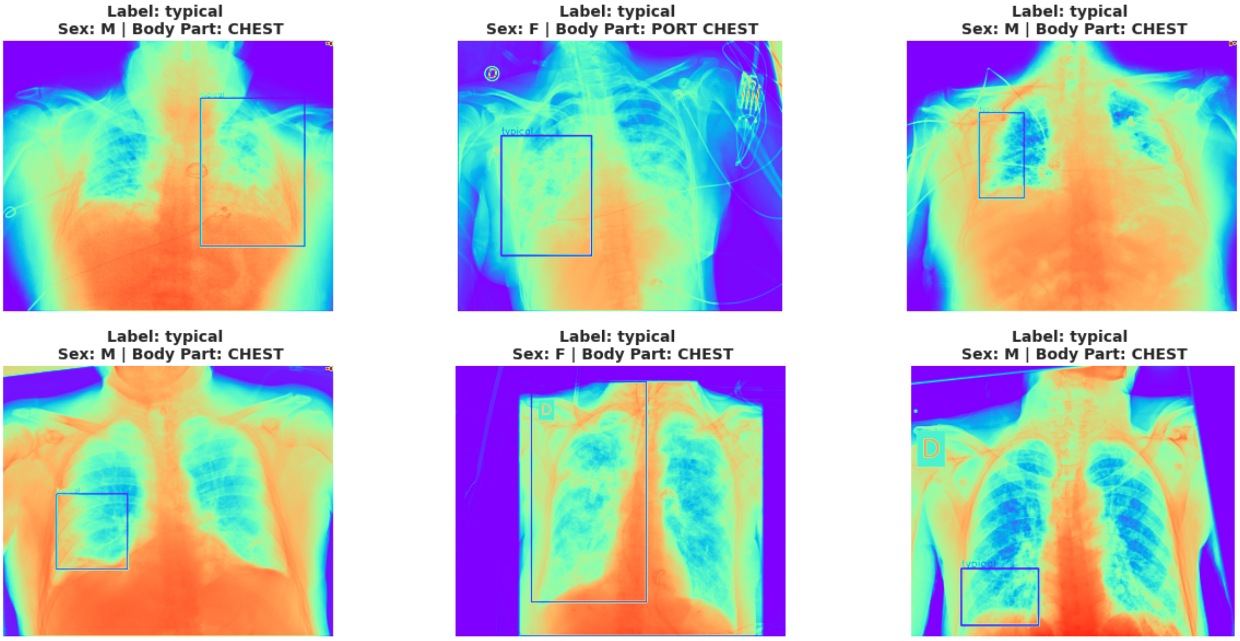
In [22]:

```
def show_dcm_info(study_ids, df):
    '''Show .dcm images along with description.'''
    wandb_logs = []

    fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(21,10))

    # Get .dcm paths
    dcm_paths = [glob.glob(f"../input/siim-covid19-detection/train/{study_id}/*/*")[0]
                 for study_id in study_ids]
    datasets = [pydicom.dcmread(path) for path in dcm_paths]
    images = [apply_voi_lut(dataset.pixel_array, dataset) for dataset in datasets]

    # Loop through the information
    for study_id, data, img, i in zip(study_ids, datasets, images, range(2*3)):
        # Fix inverted images
        img = fix_inverted_radiograms(data, img)
        # Below function available in functions section ;)
        label, bbox = get_image_metadata(study_id, df)

        # Check for bounding box and add if it's the case
        try:
            # For no bbox, the list is [nan]
            no_box = math.isnan(bbox[0])
            pass
        except TypeError:
            # Retrieve the bounding box
            all_coords = []
            for box in bbox:
                all_coords.append(return_coords(box))

            for (x1, y1, x2, y2) in all_coords:
                cv2.rectangle(img, (x1, y1), (x2, y2), (0, 80, 255), 15)
                cv2.putText(img, label, (x1, y1-14),
                            cv2.FONT_HERSHEY_SIMPLEX, 3, (0, 0, 0), 4)

        # Plot the image
        x = i // 3
        y = i % 3

        axes[x, y].imshow(img, cmap="rainbow")
        axes[x, y].set_title(f"Label: {label} \n Sex: {data.PatientSex} | Body Part:
{data.BodyPartExamined}",
              fontsize=14, weight='bold')
        axes[x, y].axis('off');

        # Save to W&B
        wandb_logs.append(wandb.Image(img,
                    caption=f"Label: {label} \n Sex: {data.PatientSex} | Body Part:
{data.BodyPartExamined}"))

  wandb.log({f"{label}": wandb_logs})
```

Typical Appearance

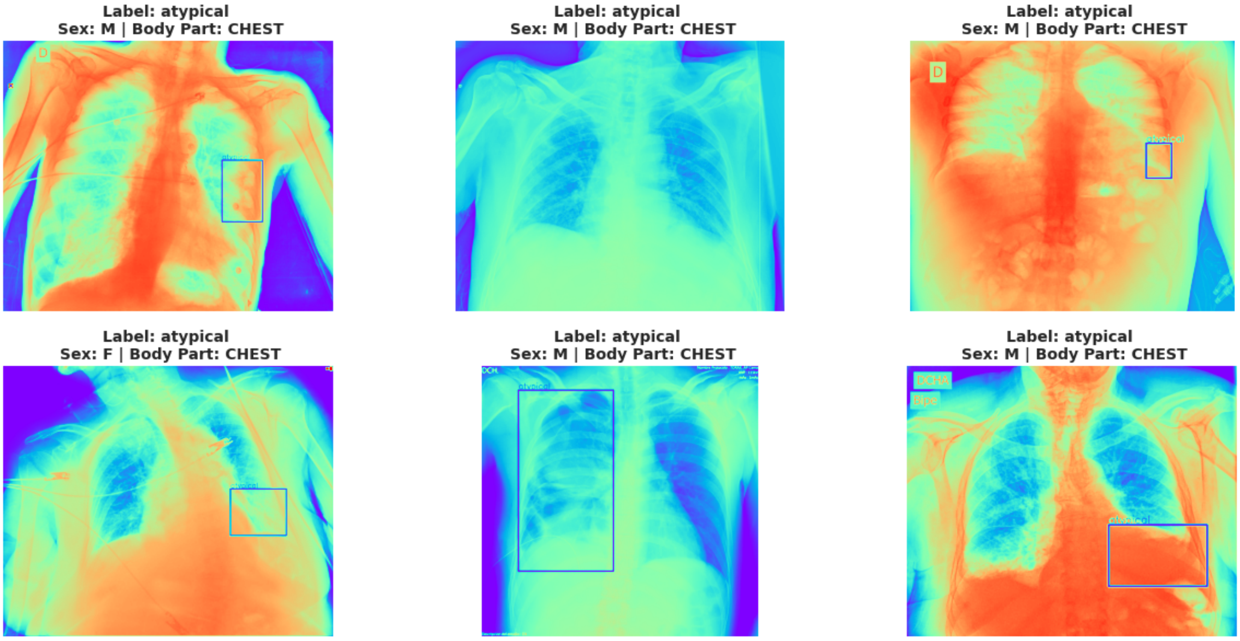
In [23]:

```
show_dcm_info(study_ids=["72044bb44d41", "5b65a69885b6", "6aa32e76f998",
                "c9ffe6312921", "082cafb03942", "d3e83031ebea"],
          df=train)
```


Label: typical
Sex: M | Body Part: CHEST
Label: typical
Sex: F | Body Part: PORT CHEST
Label: typical
Sex: M | Body Part: CHEST
Label: typical
Sex: M | Body Part: CHEST
Label: typical
Sex: F | Body Part: CHEST
Label: typical
Sex: M | Body Part: CHEST

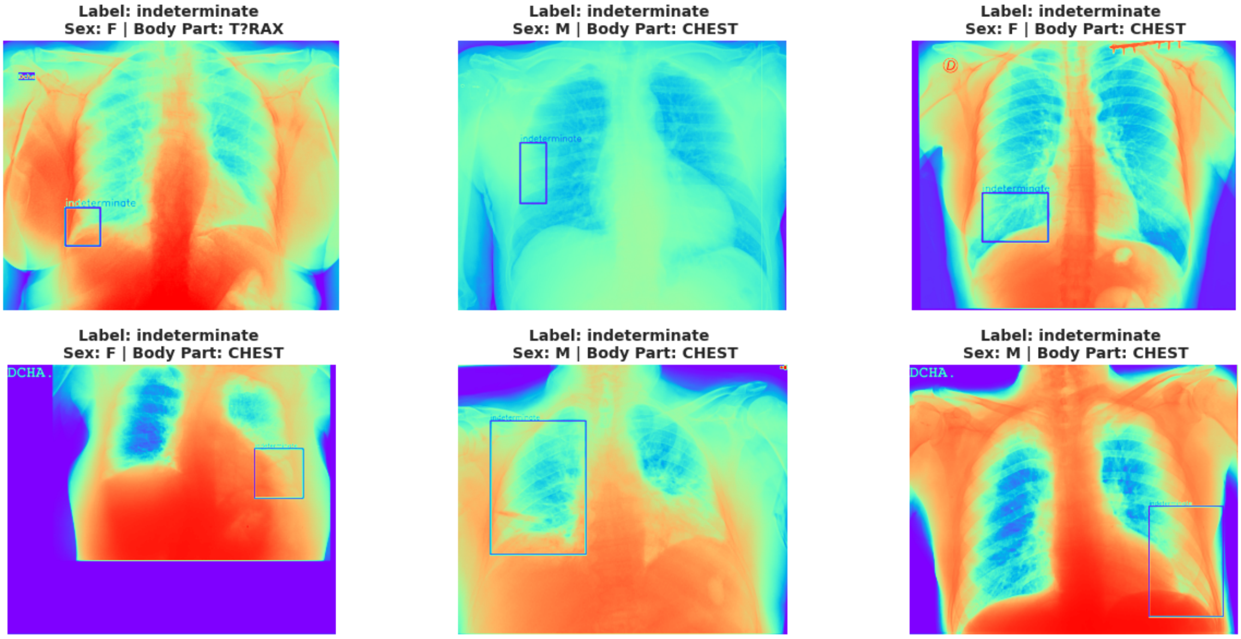
Atypical Appearance

```
In [24]:  show_dcm_info(study_ids=["f807cd855d31", "8087e3bc0efe", "7249de10ed69",
                "e300a4e86207", "4bac6c7da8b8", "f2d30ac37f7b"],
          df=train)
```


Label: atypical
Sex: M | Body Part: CHEST
Label: atypical
Sex: M | Body Part: CHEST
Label: atypical
Sex: M | Body Part: CHEST
Label: atypical
Sex: F | Body Part: CHEST
Label: atypical
Sex: M | Body Part: CHEST
Label: atypical
Sex: M | Body Part: CHEST

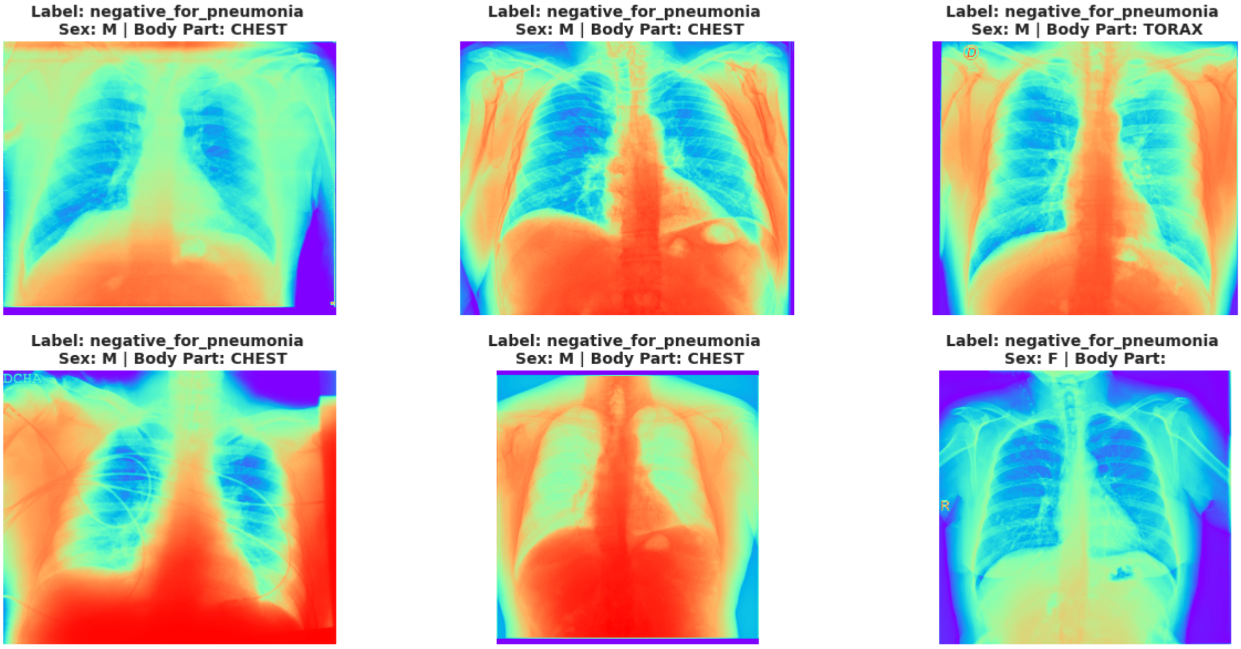
Indeterminate Appearance

```
In [25]: show_dcm_info(study_ids=["b949689a9ef1", "fe7e6015560d", "feffa20fac13",
                "747483509d0e", "c70369caef91", "1e1b4b1b53cb"], df=train)
```



Negative for Pneumonia

```
In [26]: show_dcm_info(study_ids=["612ea5194007", "db14e640e037", "d4ab797396b4",
                "6ae8a88c4b0c", "b3cf474bee3b", "0ba55e5422ab"],  df=train)
```


Label: negative_for_pneumonia
Sex: M | Body Part: CHEST
Label: negative_for_pneumonia
Sex: M | Body Part: CHEST
Label: negative_for_pneumonia
Sex: M | Body Part: TORAX
Label: negative_for_pneumonia
Sex: M | Body Part: CHEST
DCHA
Label: negative_for_pneumonia
Sex: M | Body Part: CHEST
Label: negative_for_pneumonia
Sex: F | Body Part:
R

2.2 Magic - save images & bounding boxes to W&B

In [28]:

```
# Get a few example ids (image_id)
example_ids = ["000a312787f2", "0012ff7358bc", "001398f4ff4f", "001bd15d1891",
"002e9b2128d0", "ffbeafe30b77",  "0022227f5adf", "00a129830f4e", "01376c1ba556",
"008ca392cff3"]


# Read in datas
study_ids = train[train["image_id"].isin(example_ids)]["study_id"].values
paths = [glob.glob(f"../input/siim-covid19-detection/train/{i}/*/*")[0]
        for i in study_ids]


# Retrieve resized information
images, coords, labels = [], [], []
for path, study_id in zip(paths, study_ids):
    try:
        # Read data file
        data = pydicom.dcmread(path)
        # Get image data
        img = apply_voi_lut(data.pixel_array, data)
        # Get image coordinates
        label, bbox = get_image_metadata(study_id=study_id, df=train)
        coord = [return_coords(box) for box in bbox]

        # Fix inverted radiograms + resize
        img = fix_inverted_radiograms(data, img)
        resized_img, resized_coord = resize_img_and_coord(img, coord, resize=200)

        images.append(resized_img)
        coords.append(resized_coord)
        labels.append(label)
    except RuntimeError:
        pass
```

In [29]:

```
# Map each label to a number
class_label_to_id = {'atypical': 0, 'indeterminate': 1, 'typical': 2}
# And each number to a label
class_id_to_label = {val: key for key, val in class_label_to_id.items()}

# Log each image
wandb_bbox_list = []

for image, coord, label in zip(images, coords, labels):
    wandb_bbox_list.append(wandb_bbox(image=image,
                                      bboxes=coord,
                                      true_label=class_label_to_id[label],
                                      class_id_to_label=class_id_to_label))

# Save images to W&B Dashboard
wandb.log({"radiograph": wandb_bbox_list})
```

```
print(color.BOLD + "Finished! Your Images were uploaded in your W&B Dashboard!" +
color.END)
```

Finished! Your Images were uploaded in your W&B Dashboard!

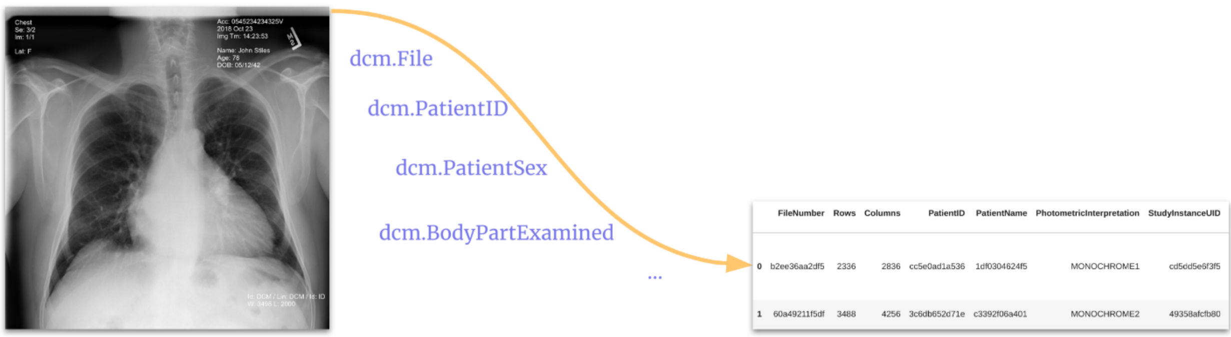
In [30]:

```
wandb.finish()
```

3. Extract Metadata from .dcm

📌 **Important**: We can create *more* **metadata** (more information on the images) from the information stored in the .dcm files. Below I am extracting all features stored in each .dcm file and storing them into a sepparate dataframe.

3.1 Store and Save Metadata



	FileNumber	Rows	Columns	PatientID	PatientName	PhotometricInterpretation	StudyInstanceUID
0	b2ee36aa2df5	2336	2836	cc5e0ad1a536	1df0304624f5	MONOCHROME1	cd5dd5e6f3f5
1	60a49211f5df	3488	4256	3c6db652d71e	c3392f06a401	MONOCHROME2	49358afcfb80

In [31]:

```
def get_observation_data(path):
    """Get information from the .dcm files.
   path: complete path to the .dcm file"""

    image_data = pydicom.read_file(path)

    # Dictionary to store the information from the image
    observation_data = {
        "FileNumber" : path.split("/")[5],
        "Rows" : image_data.get("Rows"),
        "Columns" : image_data.get("Columns"),
        "PatientID" : image_data.get("PatientID"),
        "PatientName" : image_data.get("PatientName"),
        "PhotometricInterpretation" : image_data.get("PhotometricInterpretation"),
        "StudyInstanceUID" : image_data.get("StudyInstanceUID"),
        "SamplesPerPixel" : image_data.get("SamplesPerPixel"),
        "BitsAllocated" : image_data.get("BitsAllocated"),
        "BitsStored" : image_data.get("BitsStored"),
        "HighBit" : image_data.get("HighBit"),
        "PixelRepresentation" : image_data.get("PixelRepresentation"),
    }

    # String columns
    str_columns = ["ImageType", "Modality", "PatientSex", "BodyPartExamined"]
    for k in str_columns:
        observation_data[k] = str(image_data.get(k)) if k in image_data else None


    return observation_data
```

unfold_moreShow hidden code

3.2 Let's Analyse the new information

In [33]:

```
# Save data to W&B Dashboard
save_dataset_artifact(run_name='dave-dcm-meta',
                      artifact_name='dcm_metadata',
                      path="../input/siimfisabiorsna-covid-2021/meta_train.csv")
```

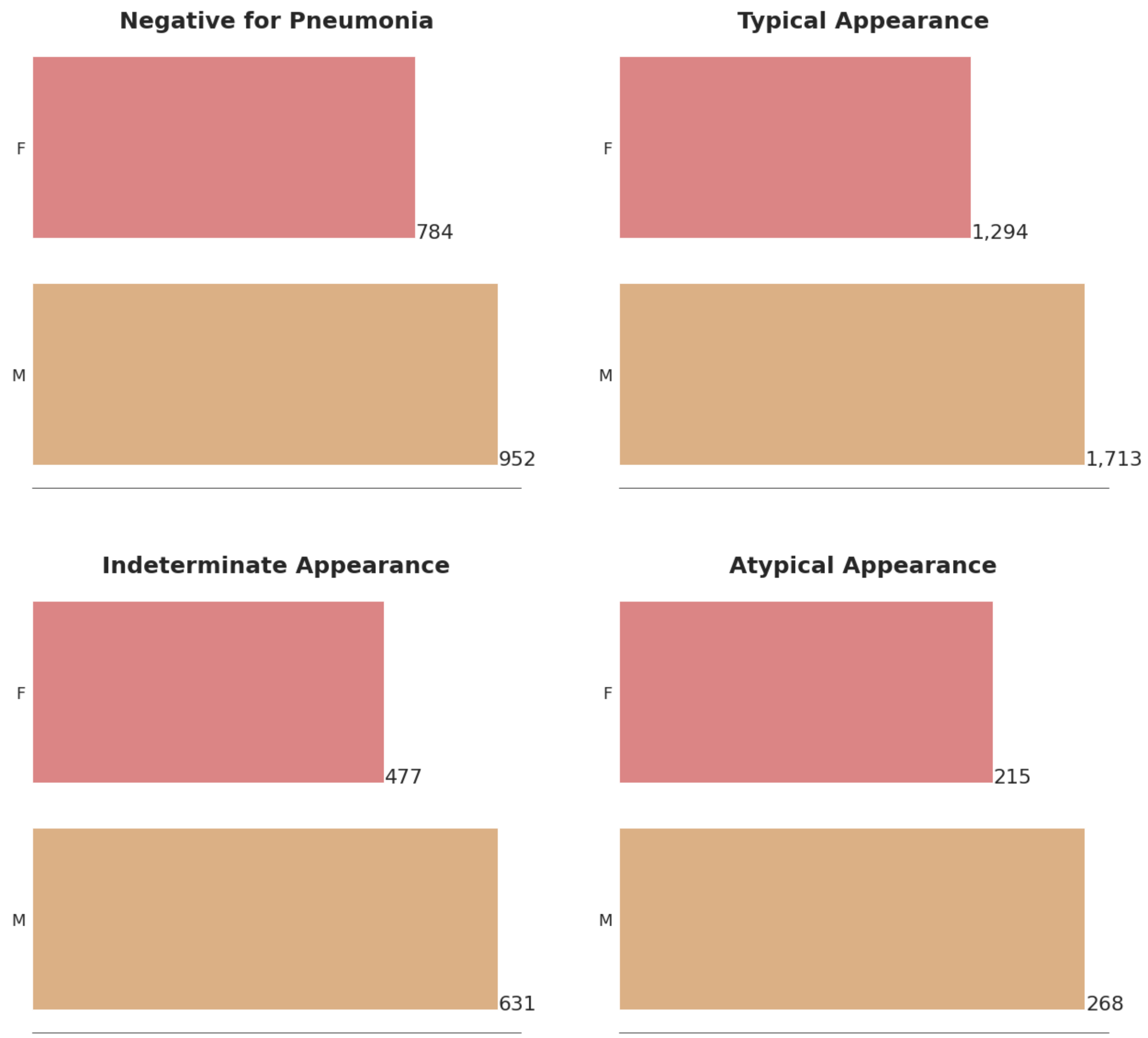
In [34]:

```
# Import
dcm_meta = pd.read_csv("../input/siimfisabiorsna-covid-2021/meta_train.csv")
dcm_meta = pd.concat([dcm_meta, train], axis=1)

dcm_meta.head(2)
```

Patient's Gender

In [35]:

```
# Get the Data
labels = ['Negative for Pneumonia', 'Typical Appearance',
          'Indeterminate Appearance', 'Atypical Appearance']
dt = dcm_meta.groupby("PatientSex")[labels].sum().reset_index()

# Plotting
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, figsize=(21,20))
axs = [ax1, ax2, ax3, ax4]

for ax, title in zip(axs, labels):
    sns.barplot(data=dt, x=title, y="PatientSex",
                ax=ax, palette=my_colors[0:])
    ax.set_title(title, fontsize=25, weight='bold')
    show_values_on_bars(ax, h_v="h", space=0.4)
    ax.set_xticklabels([])
    ax.set_ylabel('')
    ax.set_xlabel('')
```

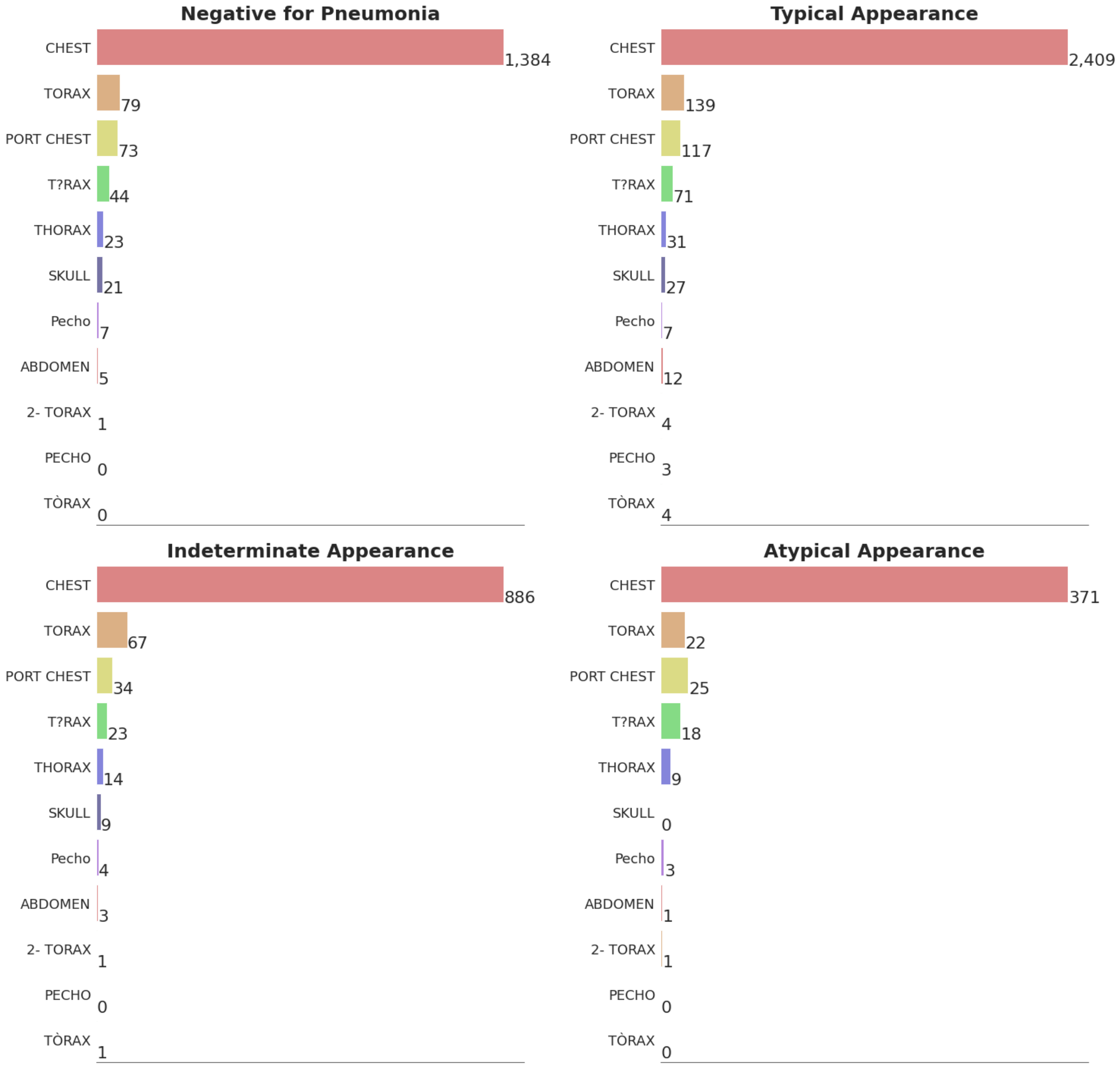


Location of X-rays

In [36]:

```
# Get the Data
labels = ['Negative for Pneumonia', 'Typical Appearance',
          'Indeterminate Appearance', 'Atypical Appearance']
dt = dcm_meta.groupby("BodyPartExamined")[labels].sum().reset_index()
dt = dt.sort_values("Negative for Pneumonia", ascending=False)

# Plotting
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, figsize=(21,20))
axs = [ax1, ax2, ax3, ax4]
```

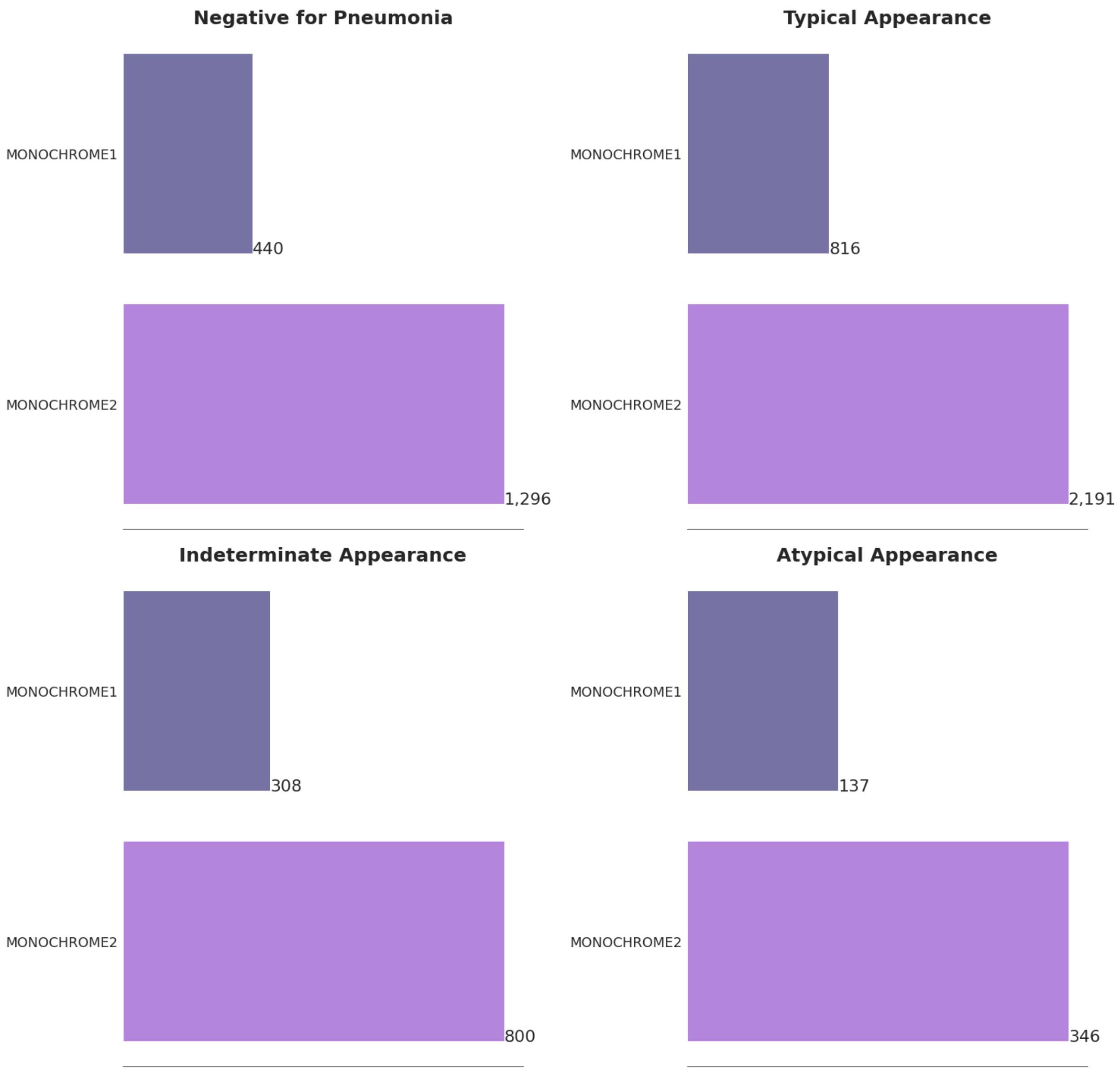
```
for ax, title in zip(axs, labels):
    sns.barplot(data=dt, x=title, y="BodyPartExamined",
                ax=ax, palette=my_colors)
    ax.set_title(title, fontsize=25, weight='bold')
    show_values_on_bars(ax, h_v="h", space=0.4)
    ax.set_xticklabels([])
    ax.set_ylabel('')
    ax.set_xlabel('')
    fig.tight_layout()
```



MONOCHROME1 or MONOCHROME2? That's the question ...

In [37]:

```
# Get the Data
labels = ['Negative for Pneumonia', 'Typical Appearance',
          'Indeterminate Appearance', 'Atypical Appearance']
dt = dcm_meta.groupby("PhotometricInterpretation")[labels].sum().reset_index()

# Plotting
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, figsize=(21,20))
axs = [ax1, ax2, ax3, ax4]

for ax, title in zip(axs, labels):
    sns.barplot(data=dt, x=title, y="PhotometricInterpretation",
                ax=ax, palette=my_colors[5:])
    ax.set_title(title, fontsize=25, weight='bold')
    show_values_on_bars(ax, h_v="h", space=0.4)
    ax.set_xticklabels([])
    ax.set_ylabel('')
    ax.set_xlabel('')
    fig.tight_layout()
```


Negative for Pneumonia
MONOCHROME1
440
MONOCHROME2
1,296
Typical Appearance
MONOCHROME1
816
MONOCHROME2
2,191
Indeterminate Appearance
MONOCHROME1
308
MONOCHROME2
800
Atypical Appearance
MONOCHROME1
137
MONOCHROME2
346

REFERENCES

- *BIMCV COVID-19+: a large annotated dataset of RX and CT images from COVID-19 patients*

- *The RSNA International COVID-19 Open Radiology Database (RICORD)*

- https://doi.org/10.31219/osf.io/532ek

- https://docs.wandb.ai/

- https://siim.org/news/580751/SIIM-FISABIO--RSNA-Recognize-Winners-of-the-COVID-19-Detection--Localization-Challenge-on-Kaggle.htm