DataPreprocessing-EDA

April 13, 2025

Original Data source https://nihcc.app.box.com/v/ChestXray-NIHCC

Google Healthcare APIs https://cloud.google.com/healthcare-api/docs/resources/public-datasets/nih-chest

[1]:

```
!pip install kagglehub
!pip install kagglehub[pandas-datasets]
!pip install wget
!pip install keras-tuner
```

```
Requirement already satisfied: kagglehub in /usr/local/lib/python3.11/dist-
packages (0.3.11)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-
packages (from kagglehub) (24.2)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.11/dist-packages
(from kagglehub) (6.0.2)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-
packages (from kagglehub) (2.32.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages
(from kagglehub) (4.67.1)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.11/dist-packages (from requests->kagglehub) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-
packages (from requests->kagglehub) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.11/dist-packages (from requests->kagglehub) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.11/dist-packages (from requests->kagglehub) (2025.1.31)
Requirement already satisfied: kagglehub[pandas-datasets] in
/usr/local/lib/python3.11/dist-packages (0.3.11)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-
packages (from kagglehub[pandas-datasets]) (24.2)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.11/dist-packages
(from kagglehub[pandas-datasets]) (6.0.2)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-
packages (from kagglehub[pandas-datasets]) (2.32.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages
(from kagglehub[pandas-datasets]) (4.67.1)
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages
```

```
(from kagglehub[pandas-datasets]) (2.2.2)
Requirement already satisfied: numpy>=1.23.2 in /usr/local/lib/python3.11/dist-
packages (from pandas->kagglehub[pandas-datasets]) (2.0.2)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.11/dist-packages (from pandas->kagglehub[pandas-
datasets]) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-
packages (from pandas->kagglehub[pandas-datasets]) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-
packages (from pandas->kagglehub[pandas-datasets]) (2025.2)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.11/dist-packages (from requests->kagglehub[pandas-
datasets]) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-
packages (from requests->kagglehub[pandas-datasets]) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.11/dist-packages (from requests->kagglehub[pandas-
datasets]) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.11/dist-packages (from requests->kagglehub[pandas-
datasets]) (2025.1.31)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-
packages (from python-dateutil>=2.8.2->pandas->kagglehub[pandas-datasets])
(1.17.0)
Collecting wget
  Downloading wget-3.2.zip (10 kB)
  Preparing metadata (setup.py) … done
Building wheels for collected packages: wget
  Building wheel for wget (setup.py) … done
  Created wheel for wget: filename=wget-3.2-py3-none-any.whl size=9655
sha256=018f394ef0d70790c422fee28405aa5b022ca62df5c6f71b9b76aa04822c2938
  Stored in directory: /root/.cache/pip/wheels/40/b3/0f/a40dbd1c6861731779f62cc4
babcb234387e11d697df70ee97
Successfully built wget
Installing collected packages: wget
Successfully installed wget-3.2
Collecting keras-tuner
  Downloading keras_tuner-1.4.7-py3-none-any.whl.metadata (5.4 kB)
Requirement already satisfied: keras in /usr/local/lib/python3.11/dist-packages
(from keras-tuner) (3.8.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-
packages (from keras-tuner) (24.2)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-
packages (from keras-tuner) (2.32.3)
Collecting kt-legacy (from keras-tuner)
  Downloading kt_legacy-1.0.5-py3-none-any.whl.metadata (221 bytes)
Requirement already satisfied: absl-py in /usr/local/lib/python3.11/dist-
packages (from keras->keras-tuner) (1.4.0)
```

```
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages
(from keras->keras-tuner) (2.0.2)
Requirement already satisfied: rich in /usr/local/lib/python3.11/dist-packages
(from keras->keras-tuner) (13.9.4)
Requirement already satisfied: namex in /usr/local/lib/python3.11/dist-packages
(from keras->keras-tuner) (0.0.8)
Requirement already satisfied: h5py in /usr/local/lib/python3.11/dist-packages
(from keras->keras-tuner) (3.13.0)
Requirement already satisfied: optree in /usr/local/lib/python3.11/dist-packages
(from keras->keras-tuner) (0.14.1)
Requirement already satisfied: ml-dtypes in /usr/local/lib/python3.11/dist-
packages (from keras->keras-tuner) (0.4.1)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.11/dist-packages (from requests->keras-tuner) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-
packages (from requests->keras-tuner) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.11/dist-packages (from requests->keras-tuner) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.11/dist-packages (from requests->keras-tuner) (2025.1.31)
Requirement already satisfied: typing-extensions>=4.5.0 in
/usr/local/lib/python3.11/dist-packages (from optree->keras->keras-tuner)
(4.13.1)
Requirement already satisfied: markdown-it-py>=2.2.0 in
/usr/local/lib/python3.11/dist-packages (from rich->keras->keras-tuner) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in
/usr/local/lib/python3.11/dist-packages (from rich->keras->keras-tuner) (2.18.0)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-
packages (from markdown-it-py>=2.2.0->rich->keras->keras-tuner) (0.1.2)
Downloading keras_tuner-1.4.7-py3-none-any.whl (129 kB)
                         129.1/129.1 kB
3.4 MB/s eta 0:00:00
Downloading kt_legacy-1.0.5-py3-none-any.whl (9.6 kB)
Installing collected packages: kt-legacy, keras-tuner
Successfully installed keras-tuner-1.4.7 kt-legacy-1.0.5
```

0.1 Load Libraries

[2]:

```python
import os
import zipfile
import seaborn as sns
import numpy as np
import kagglehub
from kagglehub import KaggleDatasetAdapter
import pandas as pd
import matplotlib.pyplot as plt
import cv2
import urllib.request

import tensorflow as tf
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.utils.class_weight import compute_class_weight
from tensorflow.keras import layers, models, Input, Model
from kerastuner import HyperModel
from kerastuner.tuners import RandomSearch
```

```
<ipython-input-2-b7ab669529a1>:17: DeprecationWarning: `import kerastuner` is
deprecated, please use `import keras_tuner`.
  from kerastuner import HyperModel
```

[3]:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[4]:

```
# Global flags
SKIP_BOUNDING_BOX = True
SKIP_DOWNLOAD = False
SKIP_UNZIP = False

DRIVE_PATH = "/content/drive/MyDrive/AAI-590_Collabs"
RESIZED_IMAGES_ZIP_PATH = "/content/drive/MyDrive/AAI-590_Collabs"
RESIZED_IMAGES_PATH = "/content/images_resized/images_resized";
```

[5]:

```
SKIP_DOWNLOAD = os.path.exists(RESIZED_IMAGES_ZIP_PATH)
SKIP_UNZIP = os.path.exists(RESIZED_IMAGES_PATH)
```

[6]:

```
# print current variables
print("SKIP_DOWNLOAD: ", SKIP_DOWNLOAD)
print("SKIP_UNZIP: ", SKIP_UNZIP)
```

```
SKIP_DOWNLOAD:  True
SKIP_UNZIP:  False
```

0.2 Load Dataset

[7]:

```
# Set the dataset path
dataset_name = "nih-chest-xrays/data"
version = 3
# Set the path to the file you'd like to load
file_path = "Data_Entry_2017.csv"
file_path_bbox = "BBox_List_2017.csv"

gcloud_url_base = 'https://storage.googleapis.com/
  ↪gcs-public-data--healthcare-nih-chest-xray/png/'
```

[8]:

```
# Load the latest version
df = kagglehub.load_dataset(
  KaggleDatasetAdapter.PANDAS,
  dataset_name,
  file_path,
  # Provide any additional arguments like
  # sql_query or pandas_kwargs. See the
  # documenation for more information:
  # https://github.com/Kaggle/kagglehub/blob/main/README.
 ↪md#kaggledatasetadapterpandas
)

df_box_list = kagglehub.load_dataset(
  KaggleDatasetAdapter.PANDAS,
  dataset_name,
  file_path_bbox
)
```

```
<ipython-input-8-1e51267dc0e1>:2: DeprecationWarning: load_dataset is deprecated
and will be removed in future version.
  df = kagglehub.load_dataset(

Downloading from https://www.kaggle.com/api/v1/datasets/download/nih-chest-
xrays/data?dataset_version_number=3&file_name=Data_Entry_2017.csv…

100%|      | 924k/924k [00:00<00:00, 101MB/s]

Extracting zip of Data_Entry_2017.csv…


<ipython-input-8-1e51267dc0e1>:12: DeprecationWarning: load_dataset is
deprecated and will be removed in future version.
  df_box_list = kagglehub.load_dataset(

Downloading from https://www.kaggle.com/api/v1/datasets/download/nih-chest-
xrays/data?dataset_version_number=3&file_name=BBox_List_2017.csv…

100%|      | 90.2k/90.2k [00:00<00:00, 2.13MB/s]
```

[9]:

```
print(df['View Position'].value_counts())
```

```
View Position
PA    67310
AP    44810
Name: count, dtype: int64
```

[10]:

```
# keep orignal dataframe for reference
df_locked = df.copy()
```

0.3 Remove all where "View Position" column value is "AP"

AP means "anteroposterior dimension" which is an X-ray from front-to-back This wil affect the training with both back-to-front and front-to-back images of MRIs

[11]:

```
# Entries before removal
print(f"Before 'AP' removal: {df['View Position'].value_counts()}")

# Entries after removal
df = df[df['View Position'] != 'AP']

# Remaining data is 66.57% of total initial data
print(f"After 'AP' removal: {df['View Position'].value_counts()}")
```

```
Before 'AP' removal: View Position
PA    67310
AP    44810
Name: count, dtype: int64
After 'AP' removal: View Position
PA    67310
Name: count, dtype: int64
```

[12]:

```
links = [
    "https://nihcc.box.com/shared/static/vfk49d74nhbxq3nqjg0900w5nvkorp5c.gz",
    "https://nihcc.box.com/shared/static/i28rlmbvmfjbl8p2n3ril0pptcmcu9d1.gz",
    "https://nihcc.box.com/shared/static/f1t00wrtdk94satdfb9olcolqx20z2jp.gz",
    "https://nihcc.box.com/shared/static/0aowwzs5lhjrceb3qp67ahp0rd1l1etg.gz",
    "https://nihcc.box.com/shared/static/v5e3goj22zr6h8tzualxfsqlqaygfbsn.gz",
    "https://nihcc.box.com/shared/static/asi7ikud9jwnkrnkj99jnpfkjdes7l6l.gz",
    "https://nihcc.box.com/shared/static/jn1b4mw4n6lnh74ovmcjb8y48h8xj07n.gz",
    "https://nihcc.box.com/shared/static/tvpxmn7qyrgl0w8wfh9kqfjskv6nmm1j.gz",
    "https://nihcc.box.com/shared/static/upyy3ml7qdumlgk2rfcvlb9k6gvqq2pj.gz",
    "https://nihcc.box.com/shared/static/l6nilvfa9cg3s28tqv1qc1olm3gnz54p.gz",
    "https://nihcc.box.com/shared/static/hhq8fkdgvcari67vfhs7ppg2w6ni4jze.gz",
    "https://nihcc.box.com/shared/static/ioqwiy20ihqwyr8pf4c24eazhh281pbu.gz",
]
```

[13]:

```
# Create a dictionary for folder locations
folder_ranges = {
    "images_001": (0, 4998),  # Adjusted to 0-based index
    "images_002": (4999, 14998),
    "images_003": (14999, 24998),
    "images_004": (24999, 34998),
    "images_005": (34999, 44998),
    "images_006": (44999, 54998),
    "images_007": (54999, 64998),
    "images_008": (64999, 74998),
    "images_009": (74999, 84998),
    "images_010": (84999, 94998),
    "images_011": (94999, 104998),
    "images_012": (104999, 112120)
}

def get_image_folder(df, image_name):
    if image_name in df["Image Index"].values:
        image_index = df[df["Image Index"] == image_name].index[0]  # Get row␣
 ↪index
        # print(f"Image {image_name} is at index {image_index}")  # Debugging␣
 ↪output

        for folder, (start, end) in folder_ranges.items():
            if start <= image_index <= end:
                return folder

    return None  # If not found
```

```
[14]: display(df.head())
      display(df.tail())
      display(df.columns)
```

```
       Image Index          Finding Labels  Follow-up #  Patient ID  \
0  00000001_000.png            Cardiomegaly            0           1
1  00000001_001.png  Cardiomegaly|Emphysema            1           1
2  00000001_002.png   Cardiomegaly|Effusion            2           1
3  00000002_000.png              No Finding            0           2
4  00000003_000.png                  Hernia            0           3

   Patient Age Patient Gender View Position  OriginalImage[Width  Height]  \
0           58              M            PA                 2682     2749
1           58              M            PA                 2894     2729
2           58              M            PA                 2500     2048
3           81              M            PA                 2500     2048
4           81              F            PA                 2582     2991

   OriginalImagePixelSpacing[x     y]  Unnamed: 11
0                       0.143  0.143          NaN
1                       0.143  0.143          NaN
2                       0.168  0.168          NaN
3                       0.171  0.171          NaN
4                       0.143  0.143          NaN

             Image Index  Finding Labels  Follow-up #  Patient ID  \
112115  00030801_001.png  Mass|Pneumonia            1       30801
112116  00030802_000.png      No Finding            0       30802
112117  00030803_000.png      No Finding            0       30803
112118  00030804_000.png      No Finding            0       30804
```

```
112119  00030805_000.png      No Finding           0       30805

        Patient Age Patient Gender View Position  OriginalImage[Width  \
112115           39              M            PA                 2048
112116           29              M            PA                 2048
112117           42              F            PA                 2048
112118           30              F            PA                 2048
112119           27              M            PA                 2048

        Height]  OriginalImagePixelSpacing[x     y]  Unnamed: 11
112115     2500                        0.168  0.168          NaN
112116     2500                        0.168  0.168          NaN
112117     2500                        0.168  0.168          NaN
112118     2500                        0.168  0.168          NaN
112119     2500                        0.171  0.171          NaN
Index(['Image Index', 'Finding Labels', 'Follow-up #', 'Patient ID',
       'Patient Age', 'Patient Gender', 'View Position', 'OriginalImage[Width',
       'Height]', 'OriginalImagePixelSpacing[x', 'y]', 'Unnamed: 11'],
      dtype='object')
```

0.4 We want to have 7 generalized classes from the original 15

Take values from “Finding Labels” and convert them into more generalized labels

[15]:

```
# Create a list to store all unique labels
all_labels = []

# Iterate over the 'Finding Labels' column
for index, row in df.iterrows():
    labels = row['Finding Labels'].split('|')
    for label in labels:
        all_labels.append(label)

# Get unique labels and print them
all_labels = list(set(all_labels))
print(f"All possible options in 'Finding Labels': {all_labels}")
```

```
All possible options in 'Finding Labels': ['Effusion', 'Pneumonia', 'Hernia',
'Infiltration', 'Atelectasis', 'Mass', 'Pneumothorax', 'Emphysema',
'Pleural_Thickening', 'No Finding', 'Consolidation', 'Fibrosis', 'Edema',
'Cardiomegaly', 'Nodule']
```

[16]:

```
def generalize_labels(label):
    if label in ['Pneumonia', 'Consolidation', 'Infiltration']:
        return 'Infection/Infiltration'
    elif label in ['Edema', 'Effusion', 'Pleural_Thickening']:
        return 'Fluid Related Issues'
    elif label in ['Atelectasis', 'Pneumothorax', 'Fibrosis', 'Emphysema']:
        return 'Lung Structure Issues'
    elif label in ['Nodule', 'Mass']:
        return 'Nodule/Mass'
    elif label == 'Cardiomegaly':
        return 'Cardiac Issues'
    elif label == 'Hernia':
        return 'Hernia'
    else:
        return label  # If we don't detect an issue 'No Finding'


df['Finding Labels'] = df['Finding Labels'].apply(lambda x: '|'.
 ↪join([generalize_labels(label) for label in x.split('|')]))

# Example:
display(df.head()) # View the updated DataFrame
```

```
        Image Index                        Finding Labels  Follow-up #  \
0  00000001_000.png                        Cardiac Issues            0
1  00000001_001.png  Cardiac Issues|Lung Structure Issues            1
2  00000001_002.png   Cardiac Issues|Fluid Related Issues            2
3  00000002_000.png                            No Finding            0
4  00000003_000.png                                Hernia            0

   Patient ID  Patient Age Patient Gender View Position  OriginalImage[Width  \
0           1           58              M            PA                 2682
1           1           58              M            PA                 2894
2           1           58              M            PA                 2500
3           2           81              M            PA                 2500
4           3           81              F            PA                 2582

   Height]  OriginalImagePixelSpacing[x     y]  Unnamed: 11
0     2749                        0.143  0.143          NaN
1     2729                        0.143  0.143          NaN
2     2048                        0.168  0.168          NaN
3     2048                        0.171  0.171          NaN
4     2991                        0.143  0.143          NaN
```

[17]:

```
display(df.head())
display(df.tail())
display(df.columns)
```

```
        Image Index                        Finding Labels  Follow-up #  \
0  00000001_000.png                        Cardiac Issues            0
1  00000001_001.png  Cardiac Issues|Lung Structure Issues            1
2  00000001_002.png   Cardiac Issues|Fluid Related Issues            2
3  00000002_000.png                            No Finding            0
4  00000003_000.png                                Hernia            0
```

```
   Patient ID  Patient Age Patient Gender View Position  OriginalImage[Width  \
0           1            58             M            PA                 2682   
1           1            58             M            PA                 2894   
2           1            58             M            PA                 2500   
3           2            81             M            PA                 2500   
4           3            81             F            PA                 2582   

   Height]  OriginalImagePixelSpacing[x     y]  Unnamed: 11  
0     2749                        0.143  0.143          NaN  
1     2729                        0.143  0.143          NaN  
2     2048                        0.168  0.168          NaN  
3     2048                        0.171  0.171          NaN  
4     2991                        0.143  0.143          NaN  
             Image Index                    Finding Labels  Follow-up #  \
112115  00030801_001.png  Nodule/Mass|Infection/Infiltration            1   
112116  00030802_000.png                          No Finding            0   
112117  00030803_000.png                          No Finding            0   
112118  00030804_000.png                          No Finding            0   
112119  00030805_000.png                          No Finding            0   

        Patient ID  Patient Age Patient Gender View Position  \
112115       30801           39              M            PA   
112116       30802           29              M            PA   
112117       30803           42              F            PA   
112118       30804           30              F            PA   
112119       30805           27              M            PA   

        OriginalImage[Width  Height]  OriginalImagePixelSpacing[x     y]  \
112115                 2048     2500                        0.168  0.168   
112116                 2048     2500                        0.168  0.168   
112117                 2048     2500                        0.168  0.168   
112118                 2048     2500                        0.168  0.168   
112119                 2048     2500                        0.171  0.171   

        Unnamed: 11  
112115          NaN  
112116          NaN  
112117          NaN  
112118          NaN  
112119          NaN  
Index(['Image Index', 'Finding Labels', 'Follow-up #', 'Patient ID',
       'Patient Age', 'Patient Gender', 'View Position', 'OriginalImage[Width',
       'Height]', 'OriginalImagePixelSpacing[x', 'y]', 'Unnamed: 11'],
      dtype='object')
```

[18]:

```
display(df.describe())
display(df.info())
```

```
        Follow-up #    Patient ID   Patient Age  OriginalImage[Width  \
count  67310.000000  67310.000000  67310.000000         67310.000000
mean       4.786317  14396.542802     47.352979          2632.590016
std        9.403191   8559.885944     16.289550           374.573816
min        0.000000      1.000000      1.000000          1143.000000
25%        0.000000   7157.250000     36.000000          2500.000000
50%        1.000000  14112.000000     49.000000          2678.000000
75%        5.000000  21117.750000     59.000000          2992.000000
max      156.000000  30805.000000    412.000000          3056.000000

            Height]  OriginalImagePixelSpacing[x            y]  Unnamed: 11
count  67310.000000                 67310.000000  67310.000000          0.0
mean    2652.208468                     0.153868      0.153868          NaN
std      396.607849                     0.017179      0.017179          NaN
min     1001.000000                     0.115000      0.115000          NaN
25%     2411.000000                     0.143000      0.143000          NaN
50%     2885.000000                     0.143000      0.143000          NaN
75%     2991.000000                     0.168000      0.168000          NaN
max     3056.000000                     0.194336      0.194336          NaN
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 67310 entries, 0 to 112119
Data columns (total 12 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   Image Index                  67310 non-null  object
 1   Finding Labels               67310 non-null  object
 2   Follow-up #                  67310 non-null  int64
 3   Patient ID                   67310 non-null  int64
 4   Patient Age                  67310 non-null  int64
 5   Patient Gender               67310 non-null  object
 6   View Position                67310 non-null  object
 7   OriginalImage[Width          67310 non-null  int64
 8   Height]                      67310 non-null  int64
 9   OriginalImagePixelSpacing[x  67310 non-null  float64
 10  y]                           67310 non-null  float64
 11  Unnamed: 11                  0 non-null      float64
dtypes: float64(3), int64(5), object(4)
memory usage: 6.7+ MB
```

```
None
```

[19]:

```
display(df_box_list.head())
display(df_box_list.describe())
display(df_box_list.info())
```

```
        Image Index Finding Label     Bbox [x           y           w  \
0  00013118_008.png   Atelectasis  225.084746  547.019217   86.779661
1  00014716_007.png   Atelectasis  686.101695  131.543498  185.491525
2  00029817_009.png   Atelectasis  221.830508  317.053115  155.118644
3  00014687_001.png   Atelectasis  726.237288  494.951420  141.016949
4  00017877_001.png   Atelectasis  660.067797  569.780787  200.677966

           h]  Unnamed: 6  Unnamed: 7  Unnamed: 8
0   79.186441         NaN         NaN         NaN
1  313.491525         NaN         NaN         NaN
2  216.949153         NaN         NaN         NaN
3   55.322034         NaN         NaN         NaN
4   78.101695         NaN         NaN         NaN
```

```
          Bbox [x           y           w          h]  Unnamed: 6  Unnamed: 7  \
count  984.000000  984.000000  984.000000  984.000000         0.0         0.0
mean   398.806111  405.425364  256.334708  252.302547         NaN         NaN
std    222.700868  166.309995  167.629620  159.443635         NaN         NaN
min      5.417989   12.837934   27.306667   21.617778         NaN         NaN
25%    203.093333  293.869045  136.533333  115.674074         NaN         NaN
50%    340.249735  412.850794  214.340942  216.949153         NaN         NaN
75%    607.959365  521.641995  311.832381  367.902430         NaN         NaN
max    905.887831  876.980783  901.120000  873.379894         NaN         NaN

       Unnamed: 8
count         0.0
mean          NaN
std           NaN
min           NaN
25%           NaN
50%           NaN
75%           NaN
max           NaN
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 984 entries, 0 to 983
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Image Index    984 non-null    object
 1   Finding Label  984 non-null    object
 2   Bbox [x        984 non-null    float64
 3   y              984 non-null    float64
 4   w              984 non-null    float64
 5   h]             984 non-null    float64
 6   Unnamed: 6     0 non-null      float64
 7   Unnamed: 7     0 non-null      float64
 8   Unnamed: 8     0 non-null      float64
dtypes: float64(7), object(2)
```

```
memory usage: 69.3+ KB
```

```
None
```

[20]:

```python
# Fix column names
df_box_list = df_box_list.rename(columns={'Bbox [x': 'x', 'h]': 'h'})
df_box_list.head()
```

[20]:

```
        Image Index Finding Label           x           y           w  \
0  00013118_008.png   Atelectasis  225.084746  547.019217   86.779661
1  00014716_007.png   Atelectasis  686.101695  131.543498  185.491525
2  00029817_009.png   Atelectasis  221.830508  317.053115  155.118644
3  00014687_001.png   Atelectasis  726.237288  494.951420  141.016949
4  00017877_001.png   Atelectasis  660.067797  569.780787  200.677966

            h  Unnamed: 6  Unnamed: 7  Unnamed: 8
0   79.186441         NaN         NaN         NaN
1  313.491525         NaN         NaN         NaN
2  216.949153         NaN         NaN         NaN
3   55.322034         NaN         NaN         NaN
4   78.101695         NaN         NaN         NaN
```

0.5 EDA

[21]:

```python
# Analyze class distribution
class_counts = df['Finding Labels'].value_counts()
num_classes = df['Finding Labels'].nunique()
display(f"Number of different classes: {num_classes}")
```

```
'Number of different classes: 356'
```

One image can contain one or more labels - thus the number of classes initially shows as 836.

[22]:

```python
# Create a list to store the processed labels
all_labels = []

# Iterate over the 'Finding Labels' column
for index, row in df.iterrows():
    labels = row['Finding Labels'].split('|')
    for label in labels:
        all_labels.append(label)

all_labels = list(set(all_labels))
display(f"Number of unique labels: {len(all_labels)}")
```

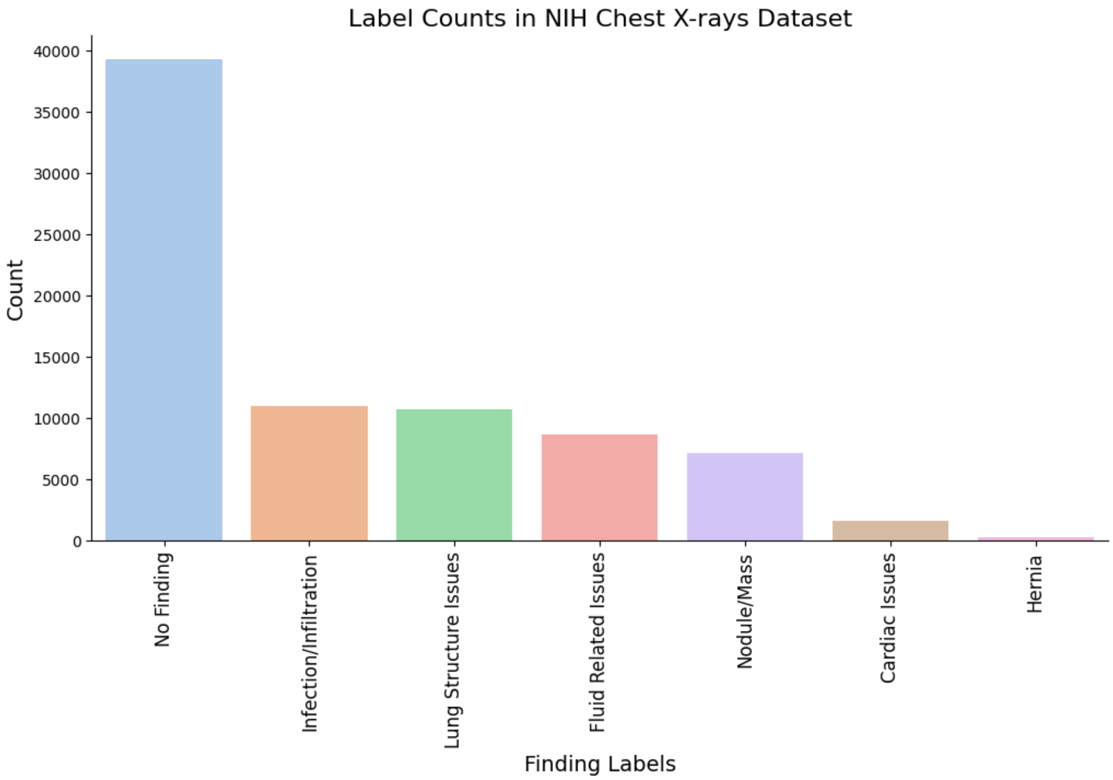
```
'Number of unique labels: 7'
```

[23]:

```python
# Count occurrences of each label
label_counts = {label: sum(df['Finding Labels'].str.contains(label)) for label␣
 ↪in all_labels}
```

```python
# Convert to DataFrame for plotting
label_counts_df = pd.DataFrame(list(label_counts.items()), columns=['Label', 'Count'])
label_counts_df = label_counts_df.sort_values(by='Count', ascending=False)

# Define a pastel color palette
palette = sns.color_palette("pastel", len(label_counts_df))

# Plot bar chart using pastel colors
plt.figure(figsize=(12, 6))
ax = sns.barplot(x='Label', y='Count', data=label_counts_df, palette=palette)

# Rotate x-axis labels
plt.xticks(rotation=90, fontsize=12)
plt.xlabel('Finding Labels', fontsize=14)
plt.ylabel('Count', fontsize=14)
plt.title('Label Counts in NIH Chest X-rays Dataset', fontsize=16)

# Remove unnecessary borders
sns.despine()
```

```
<ipython-input-23-bf94c940165c>:13: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  ax = sns.barplot(x='Label', y='Count', data=label_counts_df, palette=palette)
```



[24]:
```
display(label_counts_df)
```

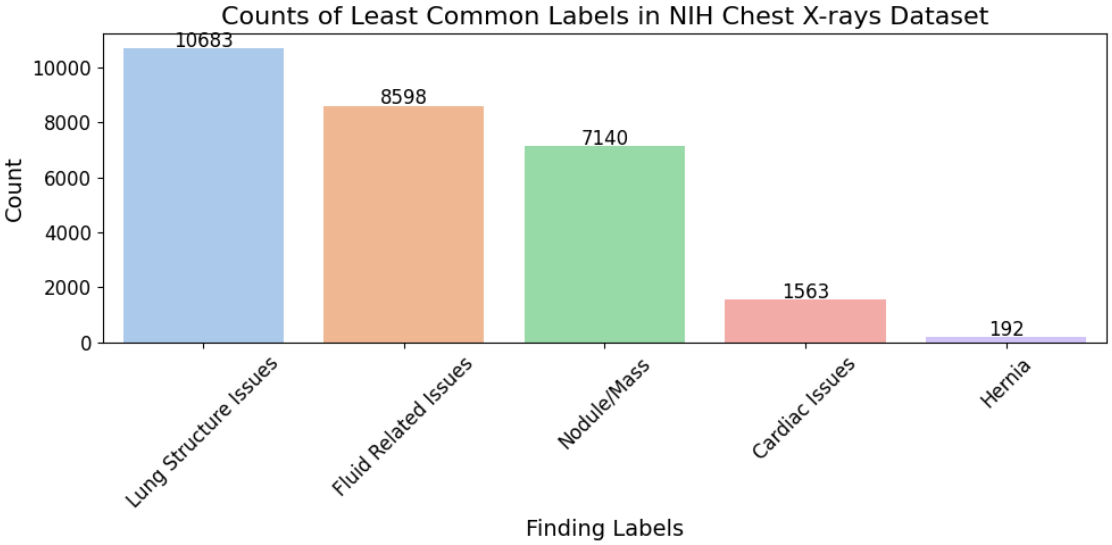
```
                    Label  Count
6              No Finding  39302
1  Infection/Infiltration  10936
2   Lung Structure Issues  10683
5    Fluid Related Issues   8598
4             Nodule/Mass   7140
0          Cardiac Issues   1563
3                  Hernia    192
```

[25]:
```
# Count occurrences of each label
label_counts = {label: sum(df['Finding Labels'].str.contains(label)) for label␣
 ↪in all_labels}

# Convert to DataFrame and sort by count
label_counts_df = pd.DataFrame(list(label_counts.items()), columns=['Label',␣
 ↪'Count'])
label_counts_df = label_counts_df.sort_values(by='Count', ascending=False)

# Extract the 5 least common labels
least_common_labels_df = label_counts_df.tail(5)
```

```
# Count occurrences for these labels in the dataset
least_common_entries = df[df['Finding Labels'].apply(lambda x: any(label in x␣
 ↪for label in least_common_labels_df['Label']))]

# Count the occurrences of these specific labels in the dataset
least_common_label_counts = {label: sum(df['Finding Labels'].str.
 ↪contains(label)) for label in least_common_labels_df['Label']}

# Convert to DataFrame
least_common_label_counts_df = pd.DataFrame(list(least_common_label_counts.
 ↪items()), columns=['Label', 'Count'])
palette = sns.color_palette("pastel")

# Create a bar plot with a custom color scheme
plt.figure(figsize=(10, 5))
ax = sns.barplot(
    x='Label',
    y='Count',
    hue='Label',  # Assign hue to avoid warning
    data=least_common_label_counts_df,
    palette=palette,
    legend=False  # Remove unnecessary legend
)

# Add count labels above bars
for index, row in least_common_label_counts_df.iterrows():
    ax.text(index, row['Count'] + 50, str(row['Count']), color='black',␣
 ↪ha="center", fontsize=12)

# Customize plot aesthetics
plt.xticks(rotation=45, fontsize=12)
plt.yticks(fontsize=12)
plt.xlabel('Finding Labels', fontsize=14)
plt.ylabel('Count', fontsize=14)
plt.title('Counts of Least Common Labels in NIH Chest X-rays Dataset',␣
 ↪fontsize=16)

# Show the plot
plt.tight_layout()
plt.show()

# Remove top and right border for a cleaner look
sns.despine()

# Show the plot
plt.show()
```

```
# Display the number of entries that contain the least common labels
least_common_entries_count = least_common_entries.shape[0]
print(f"Number of entries that contain at least one of the five least common␣
 ↪labels: {least_common_entries_count}")
```

```
<ipython-input-25-f52b0187c395>:23: UserWarning: The palette list has more
values (10) than needed (5), which may not be intended.
  ax = sns.barplot(
```



```
<Figure size 640x480 with 0 Axes>
```

```
Number of entries that contain at least one of the five least common labels:
22003
```

0.6 Notes

- The class with lowest number of entries contains 192 images. Since it will be hard to generate synthetic data, reducing all classes to 192 images to address class imbalance will significatly reduce available data.

We can consider the following strategies for addressing the class imbalances.

1. Data Augmentation (For Underrepresented Classes)

Best for Image Data

To increase the number of images in underrepresented classes, we can apply data augmentation techniques such as:

- Geometric Transformations: Rotation, flipping, cropping, translation, scaling.
- Color Variations: Adjust brightness, contrast, saturation.
- Noise Injection: Adding Gaussian noise or slight blur.

- Generative Models: Use GANs (Generative Adversarial Networks) or Diffusion Models to synthesize realistic chest X-ray images.

2. Class Weighting in Model Training

Best for Training Deep Learning Models

Instead of balancing data at the dataset level, we can adjust the model's loss function to give higher weight to underrepresented classes.

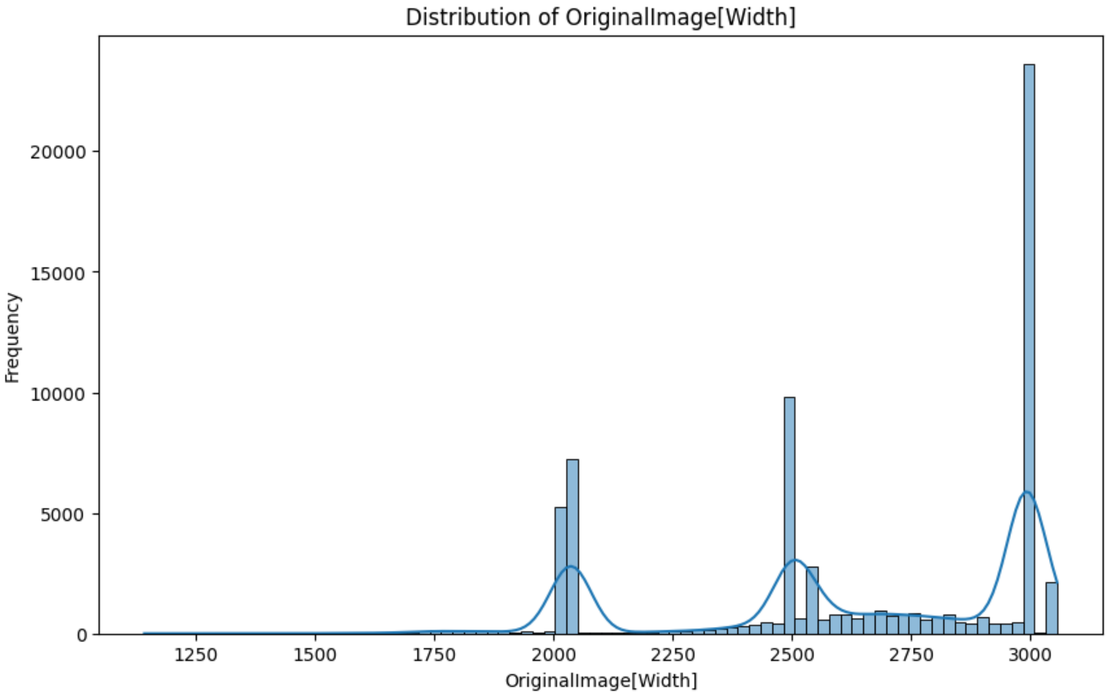
3. Oversampling the Minority Classes

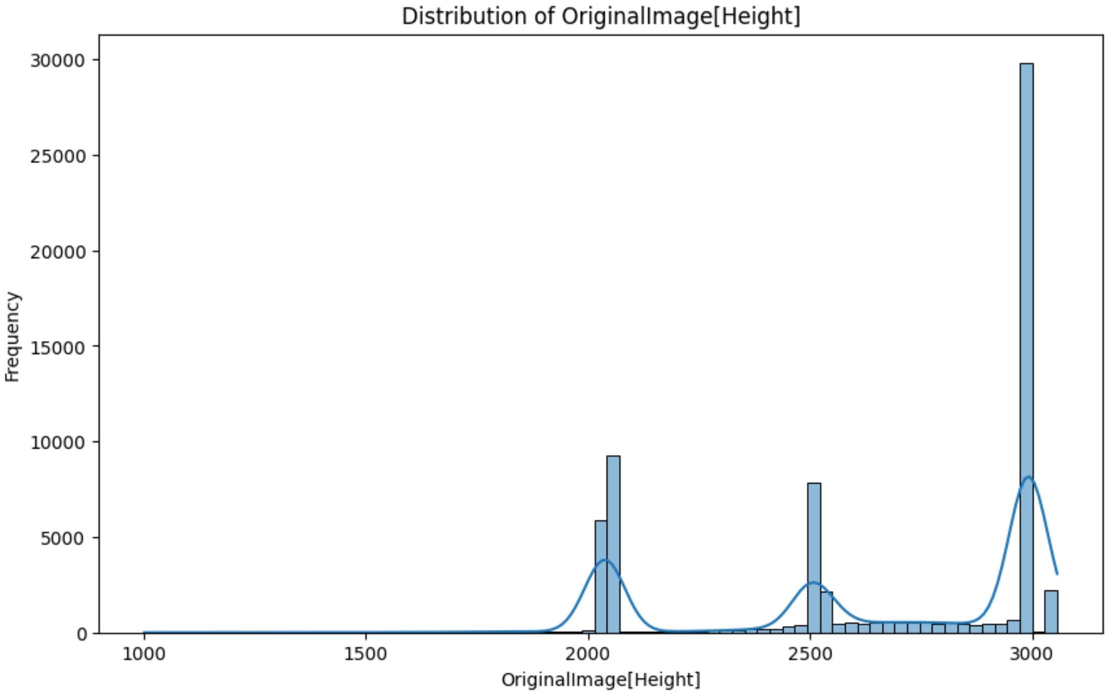
Best when Augmentation is not feasible

Instead of generating new synthetic data, we can duplicate existing samples in the underrepresented classes.

[26]:

```
# Plot the distribution of OriginalImage[Width]
plt.figure(figsize=(10, 6))
sns.histplot(df['OriginalImage[Width'], kde=True)
plt.title('Distribution of OriginalImage[Width]')
plt.xlabel('OriginalImage[Width]')
plt.ylabel('Frequency')
plt.show()

# Plot the distribution of OriginalImage[Height]
plt.figure(figsize=(10, 6))
sns.histplot(df['Height]'], kde=True)
plt.title('Distribution of OriginalImage[Height]')
plt.xlabel('OriginalImage[Height]')
plt.ylabel('Frequency')
plt.show()
```


Distribution of OriginalImage[Width]
Frequency
0
5000
10000
15000
20000
1250
1500
1750
2000
2250
2500
2750
3000
OriginalImage[Width]


Distribution of OriginalImage[Height]
Frequency
0
5000
10000
15000
20000
25000
30000
1000
1500
2000
2500
3000
OriginalImage[Height]

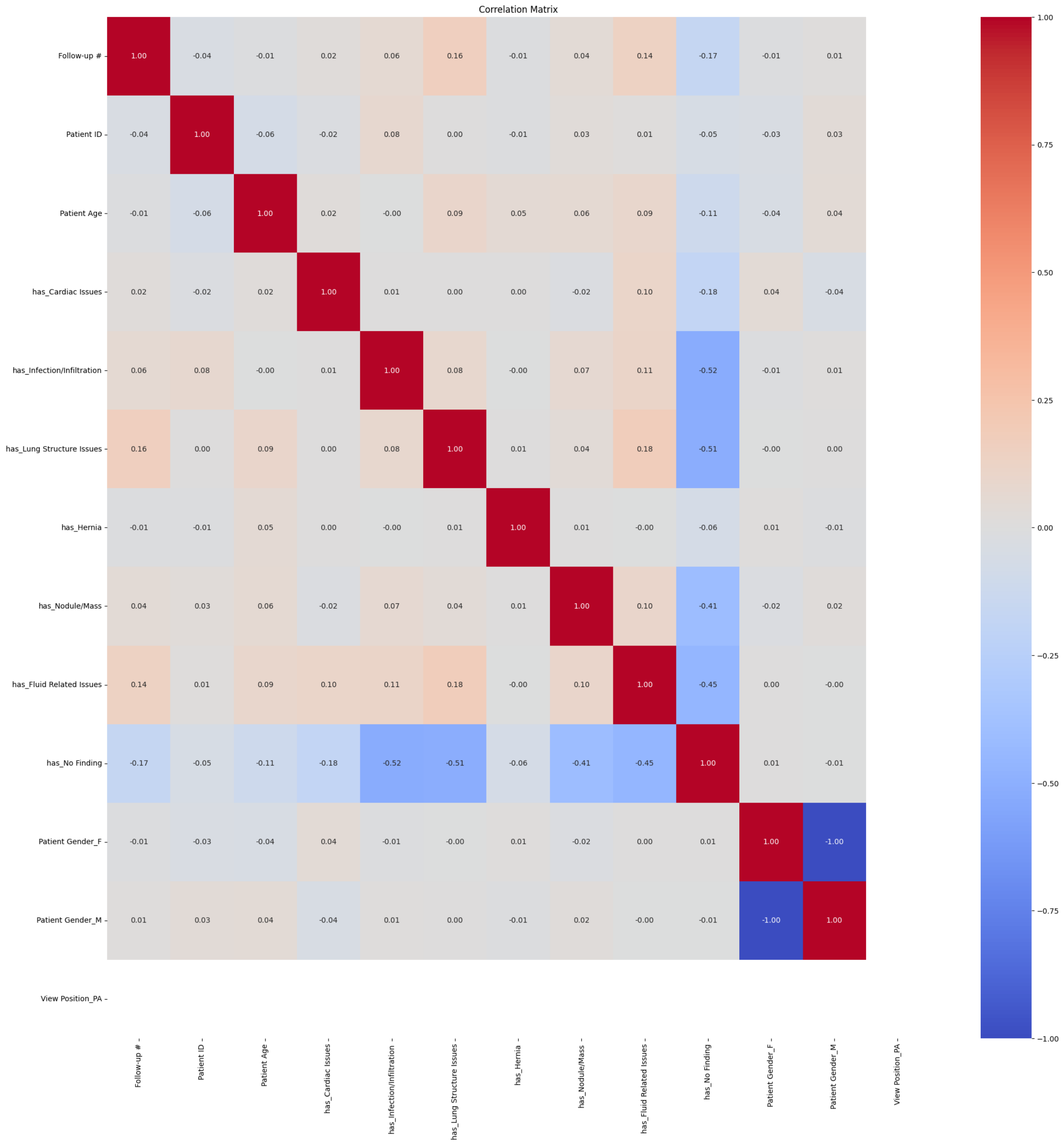
0.7 Correlation Matrix

[27]:

```
# Create a copy of the DataFrame to avoid modifying the original
df_processed = df.copy()

# Extract each label to a separate boolean column
for label in all_labels:
  df_processed[f'has_{label}'] = df_processed['Finding Labels'].str.
 ↪contains(label)

# drop Finding Labels, Image Index, 'OriginalImage[Width', 'Height]',␣
 ↪'OriginalImagePixelSpacing[x', 'y]', 'Unnamed: 11'
df_processed = df_processed.drop(columns=['Finding Labels', 'Image Index',␣
 ↪'OriginalImage[Width', 'Height]', 'OriginalImagePixelSpacing[x', 'y]',␣
 ↪'Unnamed: 11'])

display(df_processed.head())
# Encode categorical columns using one-hot encoding
categorical_columns = ['Patient Gender', 'View Position']
for column in categorical_columns:
  df_processed = pd.get_dummies(df_processed, columns=[column], prefix=[column])

# Encode the label for the boolean columns
for label in all_labels:
  df_processed[f'has_{label}'] = df_processed[f'has_{label}'].astype(int)


# Build the correlation matrix
correlation_matrix = df_processed.corr()



# Plot correlation matrix
plt.figure(figsize=(25, 25))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix')
plt.show()
```

```
   Follow-up #  Patient ID  Patient Age Patient Gender View Position  \
0            0           1           58              M            PA
1            1           1           58              M            PA
2            2           1           58              M            PA
3            0           2           81              M            PA
4            0           3           81              F            PA

   has_Cardiac Issues  has_Infection/Infiltration  has_Lung Structure Issues  \
0                True                       False                      False
1                True                       False                       True
2                True                       False                      False
```

```
3              False                        False                         False
4              False                        False                         False

   has_Hernia  has_Nodule/Mass  has_Fluid Related Issues  has_No Finding
0       False            False                     False           False
1       False            False                     False           False
2       False            False                      True           False
3       False            False                     False            True
4        True            False                     False           False
```



	Follow-up #	Patient ID	Patient Age	has_Cardiac Issues	has_Infection/Infiltration	has_Lung Structure Issues	has_Hernia	has_Nodule/Mass	has_Fluid Related Issues	has_No Finding	Patient Gender_F	Patient Gender_M	View Position_PA
Follow-up #	1.00	-0.04	-0.01	0.02	0.06	0.16	-0.01	0.04	0.14	-0.17	-0.01	0.01	
Patient ID	-0.04	1.00	-0.06	-0.02	0.08	0.00	-0.01	0.03	0.01	-0.05	-0.03	0.03	
Patient Age	-0.01	-0.06	1.00	0.02	-0.00	0.09	0.05	0.06	0.09	-0.11	-0.04	0.04	
has_Cardiac Issues	0.02	-0.02	0.02	1.00	0.01	0.00	0.00	-0.02	0.10	-0.18	0.04	-0.04	
has_Infection/Infiltration	0.06	0.08	-0.00	0.01	1.00	0.08	-0.00	0.07	0.11	-0.52	-0.01	0.01	
has_Lung Structure Issues	0.16	0.00	0.09	0.00	0.08	1.00	0.01	0.04	0.18	-0.51	-0.00	0.00	
has_Hernia	-0.01	-0.01	0.05	0.00	-0.00	0.01	1.00	0.01	-0.00	-0.06	0.01	-0.01	
has_Nodule/Mass	0.04	0.03	0.06	-0.02	0.07	0.04	0.01	1.00	0.10	-0.41	-0.02	0.02	
has_Fluid Related Issues	0.14	0.01	0.09	0.10	0.11	0.18	-0.00	0.10	1.00	-0.45	0.00	-0.00	
has_No Finding	-0.17	-0.05	-0.11	-0.18	-0.52	-0.51	-0.06	-0.41	-0.45	1.00	0.01	-0.01	
Patient Gender_F	-0.01	-0.03	-0.04	0.04	-0.01	-0.00	0.01	-0.02	0.00	0.01	1.00	-1.00	
Patient Gender_M	0.01	0.03	0.04	-0.04	0.01	0.00	-0.01	0.02	-0.00	-0.01	-1.00	1.00	
View Position_PA													

[28]:

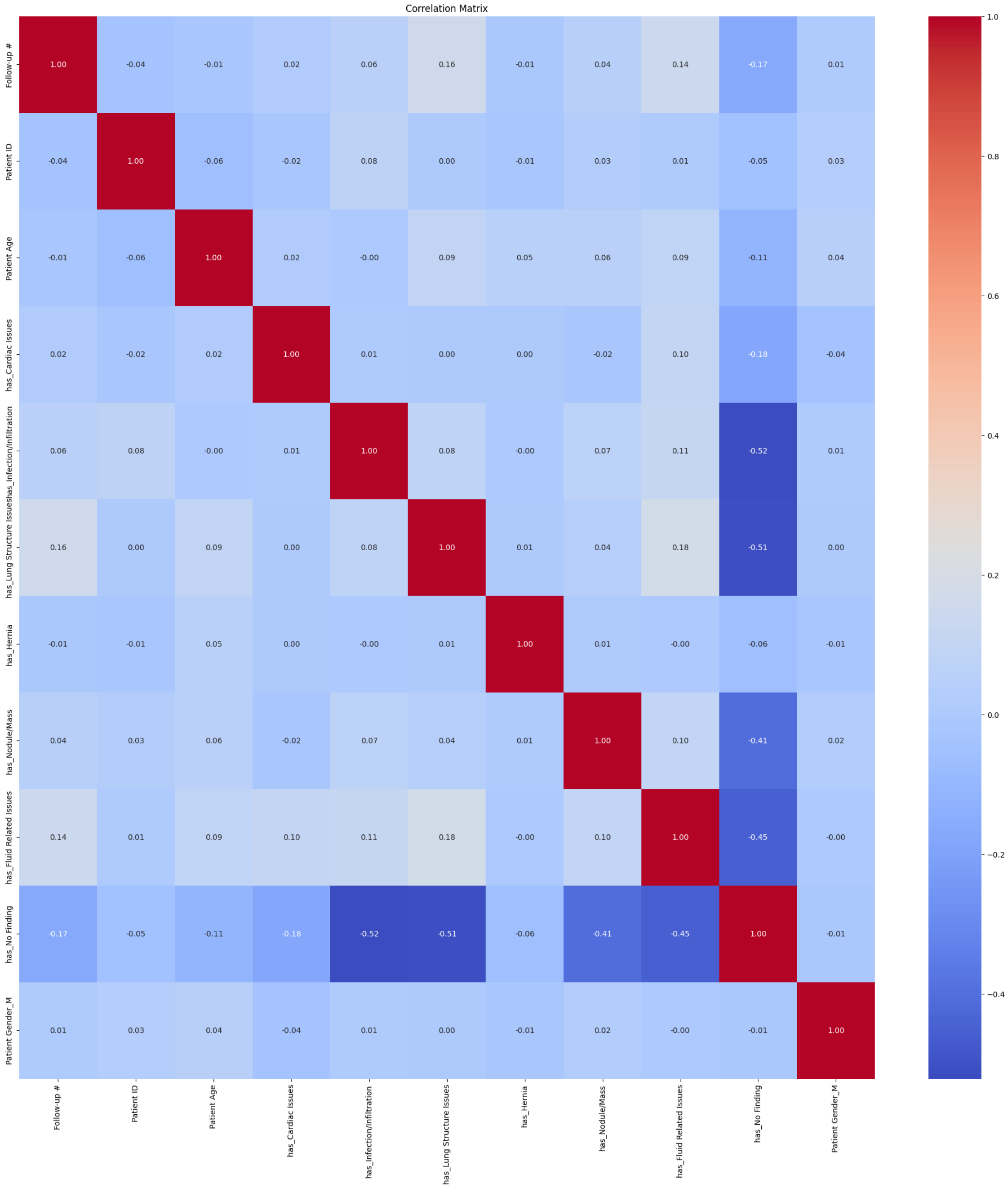
```
display(correlation_matrix)
```

```
                            Follow-up #  Patient ID  Patient Age  \
Follow-up #                    1.000000   -0.037928    -0.013289
Patient ID                    -0.037928    1.000000    -0.063590
Patient Age                   -0.013289   -0.063590     1.000000
has_Cardiac Issues             0.018726   -0.019187     0.015911
has_Infection/Infiltration     0.058027    0.077087    -0.001701
has_Lung Structure Issues      0.161326    0.004941     0.094682
has_Hernia                    -0.009183   -0.013524     0.051530
has_Nodule/Mass                0.044365    0.026500     0.055799
has_Fluid Related Issues       0.139019    0.013117     0.087079
has_No Finding                -0.166038   -0.051644    -0.112697
Patient Gender_F              -0.009111   -0.033435    -0.044918
Patient Gender_M               0.009111    0.033435     0.044918
View Position_PA                    NaN         NaN          NaN

                            has_Cardiac Issues  has_Infection/Infiltration  \
Follow-up #                           0.018726                    0.058027
Patient ID                           -0.019187                    0.077087
Patient Age                           0.015911                   -0.001701
has_Cardiac Issues                    1.000000                    0.007503
has_Infection/Infiltration            0.007503                    1.000000
has_Lung Structure Issues             0.002681                    0.079501
has_Hernia                            0.002851                   -0.000902
has_Nodule/Mass                      -0.020437                    0.067088
has_Fluid Related Issues              0.104129                    0.110167
has_No Finding                       -0.182645                   -0.521742
Patient Gender_F                      0.039572                   -0.012901
Patient Gender_M                     -0.039572                    0.012901
View Position_PA                           NaN                         NaN

                            has_Lung Structure Issues  has_Hernia  \
Follow-up #                                  0.161326   -0.009183
Patient ID                                   0.004941   -0.013524
Patient Age                                  0.094682    0.051530
has_Cardiac Issues                           0.002681    0.002851
has_Infection/Infiltration                   0.079501   -0.000902
has_Lung Structure Issues                    1.000000    0.006501
has_Hernia                                   0.006501    1.000000
has_Nodule/Mass                              0.038921    0.007810
has_Fluid Related Issues                     0.177758   -0.001273
has_No Finding                              -0.514519   -0.063357
Patient Gender_F                            -0.002363    0.014536
Patient Gender_M                             0.002363   -0.014536
View Position_PA                                  NaN         NaN

                            has_Nodule/Mass  has_Fluid Related Issues  \
Follow-up #                        0.044365                  0.139019
Patient ID                         0.026500                  0.013117
Patient Age                          0.055799                    0.087079
has_Cardiac Issues                  -0.020437                    0.104129
has_Infection/Infiltration           0.067088                    0.110167
has_Lung Structure Issues            0.038921                    0.177758
has_Hernia                           0.007810                   -0.001273
has_Nodule/Mass                      1.000000                    0.100158
has_Fluid Related Issues             0.100158                    1.000000
has_No Finding                      -0.408061                   -0.453317
Patient Gender_F                    -0.019791                    0.000407
Patient Gender_M                     0.019791                   -0.000407
View Position_PA                          NaN                         NaN

                            has_No Finding  Patient Gender_F  \
Follow-up #                      -0.166038         -0.009111
Patient ID                       -0.051644         -0.033435
Patient Age                      -0.112697         -0.044918
has_Cardiac Issues               -0.182645          0.039572
has_Infection/Infiltration       -0.521742         -0.012901
has_Lung Structure Issues        -0.514519         -0.002363
has_Hernia                       -0.063357          0.014536
has_Nodule/Mass                  -0.408061         -0.019791
has_Fluid Related Issues         -0.453317          0.000407
has_No Finding                    1.000000          0.005735
Patient Gender_F                  0.005735          1.000000
Patient Gender_M                 -0.005735         -1.000000
View Position_PA                       NaN               NaN

                            Patient Gender_M  View Position_PA
Follow-up #                         0.009111               NaN
Patient ID                          0.033435               NaN
Patient Age                         0.044918               NaN
has_Cardiac Issues                 -0.039572               NaN
has_Infection/Infiltration          0.012901               NaN
has_Lung Structure Issues           0.002363               NaN
has_Hernia                         -0.014536               NaN
has_Nodule/Mass                     0.019791               NaN
has_Fluid Related Issues           -0.000407               NaN
has_No Finding                     -0.005735               NaN
Patient Gender_F                   -1.000000               NaN
Patient Gender_M                    1.000000               NaN
View Position_PA                         NaN               NaN
```

0.8 Additional Charts and EDA

[29]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

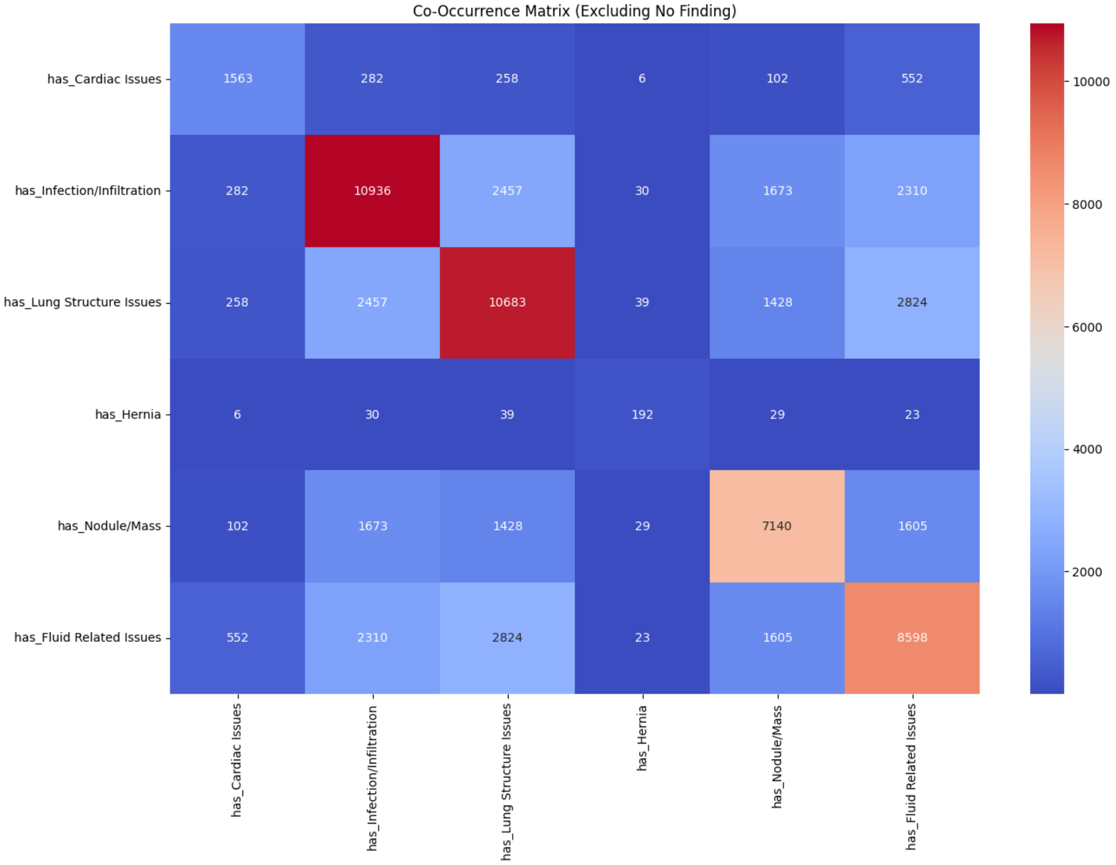
```
# Create a copy of the DataFrame to avoid modifying the original
df_processed = df.copy()

# Extract each label to a separate boolean column
for label in all_labels:
    df_processed[f'has_{label}'] = df_processed['Finding Labels'].fillna('').
  ↪str.contains(label, regex=False).astype(int)


# Drop unnecessary columns
drop_columns = ['Finding Labels', 'Image Index', 'OriginalImage[Width',␣
  ↪'Height]',
                'OriginalImagePixelSpacing[x', 'y]', 'Unnamed: 11']
df_processed = df_processed.drop(columns=drop_columns, errors='ignore')

# Encode categorical columns using one-hot encoding, dropping first category to␣
  ↪avoid redundancy
categorical_columns = ['Patient Gender', 'View Position']
df_processed = pd.get_dummies(df_processed, columns=categorical_columns,␣
  ↪drop_first=True)

# Build the correlation matrix
correlation_matrix = df_processed.corr()

# Plot correlation matrix
plt.figure(figsize=(25, 25))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix')
plt.show()
```



[30]:

```python
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

# Extract disease label columns, excluding 'No Finding'
disease_columns = [col for col in df_processed.columns if col.
 ↪startswith('has_') and col != 'has_No Finding']
```

```
# Compute the co-occurrence matrix for the remaining disease labels
co_occurrence_matrix = np.dot(df_processed[disease_columns].T,␣
↪df_processed[disease_columns])

# Convert to a DataFrame for easier visualization
co_occurrence_df = pd.DataFrame(co_occurrence_matrix, index=disease_columns,␣
↪columns=disease_columns)

# Plot the co-occurrence heatmap
plt.figure(figsize=(15, 10))
sns.heatmap(co_occurrence_df, annot=True, fmt=".0f", cmap="coolwarm")
plt.title('Co-Occurrence Matrix (Excluding No Finding)')
plt.show()
```



0.8.1 Preview Annotated Image

[31]:
```
first_image = df_box_list.iloc[0]
image_info = df_box_list[df_box_list['Image Index'] == first_image['Image␣
 ↪Index']]

display(image_info)

# Define image file name
image_name = first_image['Image Index']

# Determine the folder using dictionary lookup
image_folder = get_image_folder(df_locked, image_name)
folder_number = int(image_folder.split("_")[1])

display(f"Image Name: {image_name}")
display(image_folder)
display(folder_number)

if SKIP_BOUNDING_BOX == False:
  urllib.request.urlretrieve(links[folder_number - 1], f"{image_folder}.tar.
 ↪gz")  # download the zip file
```

```
       Image Index Finding Label           x           y          w  \
0  00013118_008.png   Atelectasis  225.084746  547.019217  86.779661

           h  Unnamed: 6  Unnamed: 7  Unnamed: 8
0  79.186441         NaN         NaN         NaN
```

```
'Image Name: 00013118_008.png'
```

```
'images_006'
```

```
6
```

[32]:
```
if SKIP_BOUNDING_BOX == False:
  !tar -xvzf images_006.tar.gz
```

[33]:
```
# Construct the image file path
image_file = f"images/{image_name}"

if SKIP_BOUNDING_BOX == False:
  try:
    image = cv2.imread(image_file)
    if image is None:
        print(f"Error: Could not read image file '{image_file}'")
    else:
        image_info = df_box_list[df_box_list['Image Index'] == image_name]
        display(image_info.columns)
        # Extract bounding box values as integers
        x = int(image_info['x'].iloc[0])
        y = int(image_info['y'].iloc[0])
        w = int(image_info['w'].iloc[0])
        h = int(image_info['h'].iloc[0])

        # Draw the bounding box on the image
        cv2.rectangle(image, (x, y), (x + w, y + h), (255, 0, 0), 4)

        # Convert BGR to RGB for displaying
        plt.figure(figsize=(10, 10))
        plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
        plt.title(f"Bounding Box on {image_name}")
        plt.axis("off")
        plt.show()
  except Exception as e:
      print(f"An error occurred: {e}")
```

0.9 Pre-processing - Image Scaling

[34]:

```
# Reduce rows with 'No Finding' label to a maximum of 10,000
no_finding_df = df[df['Finding Labels'] == 'No Finding']
if len(no_finding_df) > 10000:
    no_finding_df = no_finding_df.sample(n=10000, random_state=42)  # Randomly␣
  ↪sample 10,000 rows

# Concatenate the reduced 'No Finding' rows with other rows
other_findings_df = df[df['Finding Labels'] != 'No Finding']
df = pd.concat([no_finding_df, other_findings_df], ignore_index=True)
```

[35]:

```
len(df)
```

[35]:

```
38008
```

[36]:

```
# Rename columns
df = df.rename(columns={
    "OriginalImage[Width": "width",
    "Height]": "height",
    "OriginalImagePixelSpacing[x": "pixel_spacing x",
    "y]": "pixel_spacing y"
})

display(df.head())
```

```
        Image Index Finding Labels  Follow-up #  Patient ID  Patient Age  \
0  00019856_000.png     No Finding            0       19856           57
1  00001020_000.png     No Finding            0        1020           52
```

```
2  00008187_001.png     No Finding            1       8187            59
3  00003360_003.png     No Finding            3       3360             8
4  00014364_000.png     No Finding            0      14364            26

  Patient Gender View Position  width  height  pixel_spacing x  \
0              M            PA   2992    2991            0.143
1              M            PA   2500    2048            0.171
2              M            PA   2500    2048            0.168
3              M            PA   2048    2500            0.168
4              F            PA   2454    2991            0.143

   pixel_spacing y  Unnamed: 11
0            0.143          NaN
1            0.171          NaN
2            0.168          NaN
3            0.168          NaN
4            0.143          NaN
```

[37]:

```
# drop 'Unnamed: 11' column
df = df.drop(columns=['Unnamed: 11'], errors='ignore')
display(df.head())
```

```
        Image Index Finding Labels  Follow-up #  Patient ID  Patient Age  \
0  00019856_000.png     No Finding            0      19856            57
1  00001020_000.png     No Finding            0       1020            52
2  00008187_001.png     No Finding            1       8187            59
3  00003360_003.png     No Finding            3       3360             8
4  00014364_000.png     No Finding            0      14364            26

  Patient Gender View Position  width  height  pixel_spacing x  \
0              M            PA   2992    2991            0.143
1              M            PA   2500    2048            0.171
2              M            PA   2500    2048            0.168
3              M            PA   2048    2500            0.168
4              F            PA   2454    2991            0.143

   pixel_spacing y
0            0.143
1            0.171
2            0.168
3            0.168
4            0.143
```

[38]:

```
import os
import tarfile
import urllib.request

def download_and_extract(links, folder_ranges, df_locked):
    """Downloads image archives, extracts them, and organizes images."""

    if not os.path.exists("images"):
        os.makedirs("images")

    for i, link in enumerate(links):
        folder_name = f"images_{i+1:03d}"
        archive_name = f"{folder_name}.tar.gz"

        if not os.path.exists(os.path.join("images", archive_name)): #check if the archive already exists to prevent unnecessary downloads
          print(f"Downloading {archive_name}...")
          urllib.request.urlretrieve(link, archive_name)
        else:
          print(f"Skipping download for {archive_name} as file already exists")


        try:
            print(f"Extracting {archive_name}...")
            with tarfile.open(archive_name, "r:gz") as tar:
                tar.extractall()
            print("Extraction complete.")
        except Exception as e:
            print(f"Error extracting {archive_name}: {e}")
            continue  # Skip to the next archive if extraction fails

        # Move extracted images to the 'images' folder
        source_folder = folder_name
        if os.path.exists(source_folder):
          extracted_files = os.listdir(source_folder)
          for file in extracted_files:
            source_file = os.path.join(source_folder, file)
            destination_file = os.path.join("images", file)
            try:
              os.rename(source_file, destination_file)
            except FileExistsError:
              print(f"File {file} already exists in images folder, skipping")

          os.rmdir(source_folder)
        else:
          print(f"Folder {source_folder} doesn't exist")


        # Remove the archive file
        try:
            os.remove(archive_name)
            print(f"Removed {archive_name}")
        except OSError as e:
            print(f"Error removing {archive_name}: {e}")

if SKIP_DOWNLOAD == False:
  download_and_extract(links, folder_ranges, df_locked)
```

[39]:
```
if SKIP_DOWNLOAD == False:
  image_folder = 'images'
  num_images = len([f for f in os.listdir(image_folder) if os.path.isfile(os.
 ↪path.join(image_folder, f))])
  print(f"Number of images in '{image_folder}' folder: {num_images}")
```

[40]:
```
if SKIP_DOWNLOAD == False:
  image_folder = 'images'

  # Get a set of image names from the 'Image Index' column of the DataFrame
  image_names_in_df = set(df['Image Index'].unique())

  print(len(image_names_in_df))

  # Iterate through all files in the image folder
  for filename in os.listdir(image_folder):
      filepath = os.path.join(image_folder, filename)

      # Check if it's a file and not in the DataFrame's 'Image Index' column
      if os.path.isfile(filepath) and filename not in image_names_in_df:
          try:
              os.remove(filepath)
              print(f"Removed file: {filename}")
          except OSError as e:
              print(f"Error deleting file {filename}: {e}")
```

[41]:
```
image_folder = 'images'
def get_num_images(image_folder):
  num_images = len([f for f in os.listdir(image_folder) if os.path.isfile(os.
 ↪path.join(image_folder, f))])
  return num_images

if SKIP_DOWNLOAD == False:
  print(f"Number of images in '{image_folder}' folder:␣
 ↪{get_num_images(image_folder)}")
```

[42]:
```
if SKIP_DOWNLOAD == False:
  !python image_scale.py
```

[43]:
```
if SKIP_DOWNLOAD == False:
  !zip -r images_resized.zip images_resized
```

[44]:

```
if SKIP_DOWNLOAD == False:
  print(f"Number of images in 'images_resized' folder:␣
 ↪{get_num_images('images_resized')}")
```

[45]:

```
def zip_folder(folder_path, zip_filename):
  """Zips a folder.
  Args:
    folder_path: The path to the folder to zip.
    zip_filename: The name of the zip file to create.
  """

  # Create a zip archive
  with zipfile.ZipFile(zip_filename, 'w', zipfile.ZIP_DEFLATED) as zipf:
    for root, _, files in os.walk(folder_path):
      for file in files:
        zipf.write(os.path.join(root, file),
                   os.path.relpath(os.path.join(root, file),
                                   os.path.join(folder_path, '..')))
if SKIP_DOWNLOAD == False:
  zip_folder('images_resized', 'images_resized.zip')
```

[46]:

```
if SKIP_DOWNLOAD == False:
  !cp images_resized.zip {RESIZED_IMAGES_ZIP_PATH}
```

[47]:

```
import zipfile
import os

def unzip_files(zip_path, extract_path):
    """Unzips files from a zip archive to a specified directory.

    Args:
        zip_path: Path to the zip file.
        extract_path: Directory to extract the files to.
    """
    try:
        with zipfile.ZipFile(zip_path, 'r') as zip_ref:
            zip_ref.extractall(extract_path)
        print(f"Successfully unzipped '{zip_path}' to '{extract_path}'")
    except FileNotFoundError:
        print(f"Error: Zip file not found at '{zip_path}'")
    except zipfile.BadZipFile:
        print(f"Error: Invalid zip file at '{zip_path}'")
    except Exception as e:
        print(f"An unexpected error occurred: {e}")

# Assuming RESIZED_IMAGES_ZIP_PATH is defined and holds the correct path
if SKIP_UNZIP == False:
```

```
unzip_files(RESIZED_IMAGES_ZIP_PATH + "/images_resized.zip", "images_resized")
```

```
Successfully unzipped
'/content/drive/MyDrive/AAI-590_Collabs/images_resized.zip' to 'images_resized'
```

0.9.1 Multi-label encoding

[48]:
```
# Extract all unique labels
all_labels = sorted(set(label for sublist in df['Finding Labels'].str.
 ↪split('|') for label in sublist))
display(all_labels)

# Encode multi-labels
def encode_multilabel(labels):
    label_set = labels.split('|')
    return [1 if label in label_set else 0 for label in all_labels]

df['encoded_labels'] = df['Finding Labels'].apply(encode_multilabel)
display(df.head())

y = np.array(df['encoded_labels'].tolist())
```

```
['Cardiac Issues',
 'Fluid Related Issues',
 'Hernia',
 'Infection/Infiltration',
 'Lung Structure Issues',
 'No Finding',
 'Nodule/Mass']
```

```
        Image Index Finding Labels  Follow-up #  Patient ID  Patient Age  \
0  00019856_000.png     No Finding            0       19856           57
1  00001020_000.png     No Finding            0        1020           52
2  00008187_001.png     No Finding            1        8187           59
3  00003360_003.png     No Finding            3        3360            8
4  00014364_000.png     No Finding            0       14364           26

  Patient Gender View Position  width  height  pixel_spacing x  \
0              M            PA   2992    2991            0.143
1              M            PA   2500    2048            0.171
2              M            PA   2500    2048            0.168
3              M            PA   2048    2500            0.168
4              F            PA   2454    2991            0.143

   pixel_spacing y         encoded_labels
0            0.143  [0, 0, 0, 0, 0, 1, 0]
1            0.171  [0, 0, 0, 0, 0, 1, 0]
2            0.168  [0, 0, 0, 0, 0, 1, 0]
```

```
3             0.168  [0, 0, 0, 0, 0, 1, 0]
4             0.143  [0, 0, 0, 0, 0, 1, 0]
```

0.9.2 Encode Tabular Labels

[49]:

```
# Encode gender (e.g., Male/Female -> 0/1)
df['Patient Gender'] = LabelEncoder().fit_transform(df['Patient Gender'])

# Standardize numerical features
scaler = StandardScaler()
df['Patient Age'] = scaler.fit_transform(df[['Patient Age']])
df['Follow-up #'] = scaler.fit_transform(df[['Follow-up #']])

display(df.head())
```

```
        Image Index Finding Labels  Follow-up #  Patient ID  Patient Age  \
0  00019856_000.png     No Finding    -0.552742       19856     0.525833
1  00001020_000.png     No Finding    -0.552742        1020     0.215450
2  00008187_001.png     No Finding    -0.457542        8187     0.649986
3  00003360_003.png     No Finding    -0.267142        3360    -2.515918
4  00014364_000.png     No Finding    -0.552742       14364    -1.398540

   Patient Gender View Position  width  height  pixel_spacing x  \
0               1            PA   2992    2991            0.143
1               1            PA   2500    2048            0.171
2               1            PA   2500    2048            0.168
3               1            PA   2048    2500            0.168
4               0            PA   2454    2991            0.143

   pixel_spacing y         encoded_labels
0            0.143  [0, 0, 0, 0, 0, 1, 0]
1            0.171  [0, 0, 0, 0, 0, 1, 0]
2            0.168  [0, 0, 0, 0, 0, 1, 0]
3            0.168  [0, 0, 0, 0, 0, 1, 0]
4            0.143  [0, 0, 0, 0, 0, 1, 0]
```

### Train/Test Split

[50]:

```
# Create a copy of the DataFrame
df_processed = df.copy()

# Drop unnecessary columns
columns_to_drop = ['Finding Labels', 'width', 'height', 'pixel_spacing x',␣
 ↪'pixel_spacing y', 'View Position']
df_processed = df_processed.drop(columns=columns_to_drop, errors='ignore')

# Perform train/validation split
```

```
train_df, val_df = train_test_split(df_processed, test_size=0.2,␣
 ↪random_state=42)
print("Train size:", len(train_df))
print("Val size:", len(val_df))

display(train_df.head())
```

```
Train size: 30406
Val size: 7602

            Image Index  Follow-up #  Patient ID  Patient Age  Patient Gender  \
2855   00015255_013.png     0.684857       15255    -0.157009               1
6626   00008663_002.png    -0.362342        8663    -0.467392               0
12335  00002471_001.png    -0.457542        2471    -0.157009               1
11355  00001397_002.png    -0.362342        1397     0.339603               1
30189  00020495_000.png    -0.552742       20495     0.153374               0

              encoded_labels
2855   [0, 0, 0, 0, 0, 1, 0]
6626   [0, 0, 0, 0, 0, 1, 0]
12335  [0, 1, 0, 1, 1, 0, 0]
11355  [0, 1, 0, 0, 0, 0, 0]
30189  [0, 0, 0, 1, 0, 0, 0]
```

0.9.3 Address Class Imbalance

[51]:

```
# Flatten all labels into one array for each class
class_weights = {}
for i, label in enumerate(all_labels):
    label_values = y[:, i]
    class_weights[i] = compute_class_weight(class_weight='balanced', classes=np.
 ↪array([0, 1]), y=label_values)[1]  # 1 = positive class, classes is now a␣
 ↪NumPy array

print(class_weights)
```

```
{0: np.float64(12.15866922584773), 1: np.float64(2.2102814608048384), 2:
np.float64(98.97916666666667), 3: np.float64(1.7377468910021945), 4:
np.float64(1.7789010577553122), 5: np.float64(1.9004), 6:
np.float64(2.661624649859944)}
```

0.9.4 Data Generator (Image + Tabular)

[52]:

```
import tensorflow as tf
import os
import numpy as np

IMG_SIZE = 512  # Reduced image size
```

```
IMAGE_PATH = '/content/images_resized/images_resized'

def load_image(image_id):
    """Loads and preprocesses a single image."""
    # Convert image_id to string before joining paths
    path = os.path.join(IMAGE_PATH, image_id.numpy().decode('utf-8')) # Decodes␣
 ↪the byte string to a regular string
    image = tf.io.read_file(path)
    image = tf.image.decode_jpeg(image, channels=3)
    image = tf.image.resize(image, [IMG_SIZE, IMG_SIZE]) # Resize the image to␣
 ↪a fixed size
    image = image / 255.0  # Normalize pixel values
    return image

def make_dataset(df):
    """Creates a tf.data.Dataset for the given DataFrame."""
    image_paths = df['Image Index'].values
    tabular_features = df[['Follow-up #', 'Patient Age', 'Patient Gender']].
 ↪values.astype(np.float32)
    labels = np.array(df['encoded_labels'].tolist()).astype(np.float32)

    # Create a tf.data.Dataset from image paths and labels
    dataset = tf.data.Dataset.from_tensor_slices((image_paths,␣
 ↪tabular_features, labels))
    # Map the load_image function to load and preprocess images on the fly
    dataset = dataset.map(lambda img_path, tab_feat, label: (tf.
 ↪py_function(load_image, [img_path], tf.float32), tab_feat, label), # Use tf.
 ↪py_function
                          num_parallel_calls=tf.data.AUTOTUNE)  # Use parallel␣
 ↪calls for faster loading

    # Set the shape of the image tensor explicitly
    dataset = dataset.map(lambda image, tab_feat, label: (tf.
 ↪ensure_shape(image, [IMG_SIZE, IMG_SIZE, 3]), tab_feat, label))

    # Batch and prefetch the dataset
    dataset = dataset.batch(4).prefetch(tf.data.AUTOTUNE)
    return dataset

train_dataset = make_dataset(train_df)
val_dataset = make_dataset(val_df)
```

0.9.5 Build the Hybrid CNN + Tabular Model

[53]:

```
def build_model(hp, img_shape=(512, 512, 3), tab_shape=(3,), num_labels=len(all_labels)):
    # Image branch
    img_input = Input(shape=img_shape, name='input_layer')
    x = layers.Conv2D(hp.Int('conv_1_filters', min_value=32, max_value=128, step=32), (3, 3), activation='relu')(img_input)
    x = layers.MaxPooling2D()(x)
    x = layers.Conv2D(hp.Int('conv_2_filters', min_value=64, max_value=256, step=64), (3, 3), activation='relu')(x)
    x = layers.MaxPooling2D()(x)
    x = layers.Flatten()(x)

    # Tabular branch
    tab_input = Input(shape=tab_shape, name='input_layer_1')
    t = layers.Dense(hp.Int('dense_units', min_value=32, max_value=128, step=32), activation='relu')(tab_input)

    # Combine
    combined = layers.concatenate([x, t])
    z = layers.Dense(hp.Int('dense_units_2', min_value=64, max_value=256, step=64), activation='relu')(combined)
    z = layers.Dropout(hp.Float('dropout_rate', min_value=0.2, max_value=0.5, step=0.1))(z)
    output = layers.Dense(num_labels, activation='sigmoid')(z)

    model = Model(inputs=[img_input, tab_input], outputs=output)
    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['binary_accuracy'])
    return model
```

0.9.6 Train the Model

[54]:

```
# Custom training loop
def format_batch(batch):
    # batch is now a tuple of (image, tab, label)
    image, tab, label = batch
    return ({"input_layer": image, "input_layer_1": tab}, label)

train_ds = train_dataset.map(lambda x,y,z: format_batch((x,y,z))) # Pass all 3 elements as a single tuple to format_batch
val_ds = val_dataset.map(lambda x,y,z: format_batch((x,y,z))) # Pass all 3 elements as a single tuple to format_batch
```

[55]:

```
DRIVE_TUNER_RESULTS_DIR = DRIVE_PATH + '/tuner_results'
```

```
# Create the directory if it doesn't exist
!mkdir -p "{DRIVE_TUNER_RESULTS_DIR}"
```

```
[56]: # Create a HyperModel instance
class CustomHyperModel(HyperModel):
    def build(self, hp):
        return build_model(hp)

# Set up Keras Tuner Random Search
tuner = RandomSearch(
    CustomHyperModel(),
    objective='val_binary_accuracy',  # Optimize for validation binary accuracy
    max_trials=10,  # Number of random search trials
    executions_per_trial=3,  # Number of executions per trial
    directory=DRIVE_TUNER_RESULTS_DIR,  # Directory to save results
    project_name='image_and_tabular_tuning'
)

# Train the tuner to search for the best hyperparameters
tuner.search(train_ds, validation_data=val_ds, epochs=3,␣
 ↪class_weight=class_weights)

# Get the best model from the search
best_model = tuner.get_best_models(num_models=1)[0]

# Display the best model's summary
best_model.summary()
```

```
Trial 10 Complete [00h 37m 09s]

Best val_binary_accuracy So Far: 0.8146915833155314
Total elapsed time: 1d 01h 23m 50s

/usr/local/lib/python3.11/dist-packages/keras/src/saving/saving_lib.py:757:
UserWarning: Skipping variable loading for optimizer 'adam', because it has 2
variables whereas the saved optimizer has 22 variables.
  saveable.load_own_variables(weights_store.get(inner_path))

Model: "functional"



 Layer (type)                Output Shape                      Param #   Connected␣
↪to

 input_layer (InputLayer)    (None, 512, 512, 3)                     0   -          ␣
↪
```

```
conv2d (Conv2D)             (None, 510, 510, 128)             3,584 ␣
↪input_layer[0][0]

max_pooling2d               (None, 255, 255, 128)                 0 ␣
↪conv2d[0][0]
(MaxPooling2D)                                                                 ␣
↪

conv2d_1 (Conv2D)           (None, 253, 253, 128)           147,584 ␣
↪max_pooling2d[0][0]

max_pooling2d_1             (None, 126, 126, 128)                 0 ␣
↪conv2d_1[0][0]
(MaxPooling2D)                                                                 ␣
↪

input_layer_1               (None, 3)                             0  -         ␣
↪
(InputLayer)                                                                   ␣
↪

flatten (Flatten)           (None, 2032128)                       0 ␣
↪max_pooling2d_1[0][0]

dense (Dense)               (None, 32)                          128 ␣
↪input_layer_1[0][0]

concatenate (Concatenate)   (None, 2032160)                       0 ␣
↪flatten[0][0],
                                                                    ␣
↪dense[0][0]

dense_1 (Dense)             (None, 128)                 260,116,608 ␣
↪concatenate[0][0]

dropout (Dropout)           (None, 128)                           0 ␣
↪dense_1[0][0]

dense_2 (Dense)             (None, 7)                           903 ␣
↪dropout[0][0]


Total params: 260,268,807 (992.85 MB)

Trainable params: 260,268,807 (992.85 MB)
```

```
 Non-trainable params: 0 (0.00 B)
```

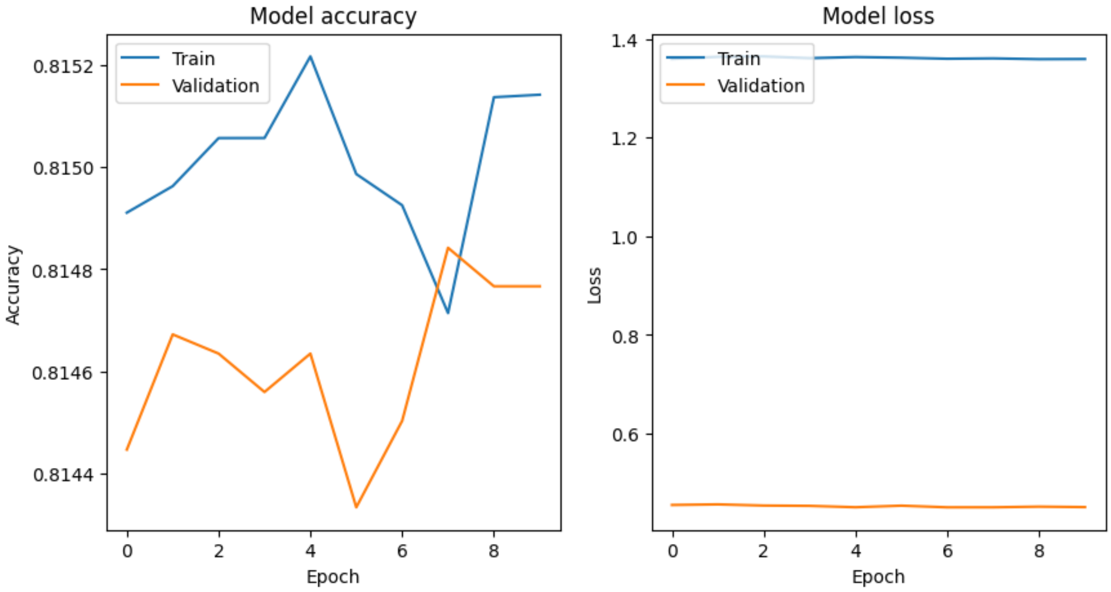
[58]:

```
# Continue training the best model for 10 epochs
best_model.fit(train_ds, validation_data=val_ds, epochs=10,␣
 ↪class_weight=class_weights)
```

```
Epoch 1/10
7602/7602                326s 43ms/step
- binary_accuracy: 0.8148 - loss: 1.3812 - val_binary_accuracy: 0.8144 -
val_loss: 0.4553
Epoch 2/10
7602/7602                300s 39ms/step
- binary_accuracy: 0.8146 - loss: 1.3894 - val_binary_accuracy: 0.8147 -
val_loss: 0.4565
Epoch 3/10
7602/7602                308s 41ms/step
- binary_accuracy: 0.8151 - loss: 1.3890 - val_binary_accuracy: 0.8146 -
val_loss: 0.4541
Epoch 4/10
7602/7602                299s 39ms/step
- binary_accuracy: 0.8150 - loss: 1.3863 - val_binary_accuracy: 0.8146 -
val_loss: 0.4535
Epoch 5/10
7602/7602                309s 41ms/step
- binary_accuracy: 0.8150 - loss: 1.3928 - val_binary_accuracy: 0.8146 -
val_loss: 0.4507
Epoch 6/10
7602/7602                296s 39ms/step
- binary_accuracy: 0.8149 - loss: 1.3933 - val_binary_accuracy: 0.8143 -
val_loss: 0.4538
Epoch 7/10
7602/7602                308s 41ms/step
- binary_accuracy: 0.8148 - loss: 1.3857 - val_binary_accuracy: 0.8145 -
val_loss: 0.4505
Epoch 8/10
7602/7602                297s 39ms/step
- binary_accuracy: 0.8146 - loss: 1.3919 - val_binary_accuracy: 0.8148 -
val_loss: 0.4506
Epoch 9/10
7602/7602                310s 41ms/step
- binary_accuracy: 0.8150 - loss: 1.3822 - val_binary_accuracy: 0.8148 -
val_loss: 0.4519
Epoch 10/10
7602/7602                299s 39ms/step
- binary_accuracy: 0.8150 - loss: 1.3900 - val_binary_accuracy: 0.8148 -
val_loss: 0.4510
```

[58]:
```
<keras.src.callbacks.history.History at 0x7d7f36bfa350>
```

[60]:
```python
import matplotlib.pyplot as plt

history = best_model.history

plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.plot(history.history['binary_accuracy'])
plt.plot(history.history['val_binary_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')

# Plot training & validation loss values
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()

# Example of plotting other metrics (if available in history.history):
if 'precision' in history.history:
    plt.plot(history.history['precision'])
    plt.plot(history.history['val_precision'])
    plt.title('Model Precision')
    plt.ylabel('Precision')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Validation'], loc='upper left')
    plt.show()

if 'recall' in history.history:
    plt.plot(history.history['recall'])
    plt.plot(history.history['val_recall'])
    plt.title('Model Recall')
    plt.ylabel('Recall')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Validation'], loc='upper left')
    plt.show()

if 'f1_score' in history.history:
    plt.plot(history.history['f1_score'])
    plt.plot(history.history['val_f1_score'])
    plt.title('Model F1 Score')
    plt.ylabel('F1 Score')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Validation'], loc='upper left')
    plt.show()
```



[63]:

```
# Save the best model to Google Drive
import os

DRIVE_MODEL_PATH = DRIVE_PATH + '/best_model-1/'  # Define the path in Google␣
 ↪Drive, adding .keras extension

# Create the directory if it doesn't exist
!mkdir -p "{DRIVE_MODEL_PATH}"

# Save the model
best_model.save(DRIVE_MODEL_PATH + 'best_model.keras')

print(f"Model saved to: {DRIVE_MODEL_PATH}")

# Verification (optional): Check if the model directory exists and is not empty
if os.path.exists(DRIVE_MODEL_PATH) and os.listdir(DRIVE_MODEL_PATH):
    print("Model directory exists and contains files.")
else:
    print("Error: Model directory does not exist or is empty.")
```

```
Model saved to: /content/drive/MyDrive/AAI-590_Collabs/best_model-1/
Model directory exists and contains files.
```