As part of continuation education and learning, I have explored Kaggle competition dataset and included insights from the classified images AI model training images. I have referenced and consolidated a variety of exploratory data analysis to better understand cervical spine fractures and their detection.

RSNA 2022 Cervical Spine Fracture Detection

Exploratory Data Analysis

Akshath Venkataraman

CONTENTS

VERTEBRAL COLUMN

The spine is made up of bones, muscles, tendons, nerves, and other tissues that reach from the base of the skull near the spinal cord (clivus) to the coccyx (tailbone). The vertebrae (back bones) of the spine include the cervical spine (C1-C7), thoracic spine (T1-T12), lumbar spine (L1-L5), sacral spine (S1-S5), and the tailbone. Each vertebra is separated by a disc. The vertebrae surround and protect the spinal cord.

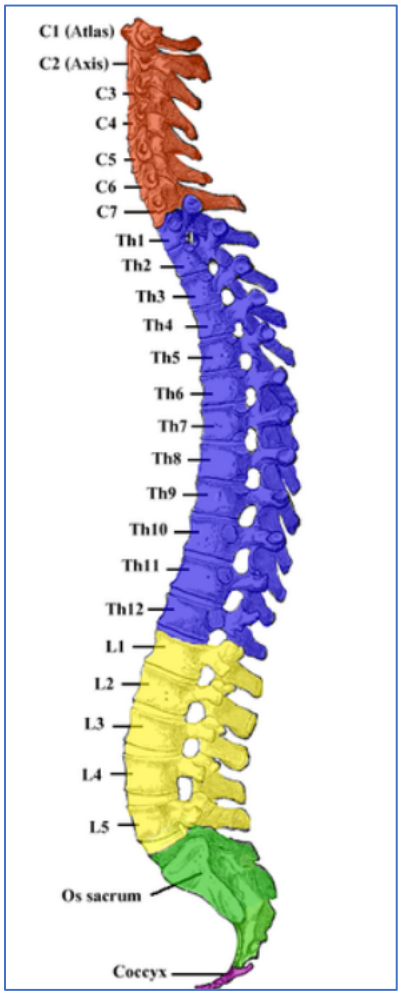


Figure A: Vertebral Column

The vertebral column encloses the spinal cord and the fluid surrounding the spinal cord. Also called backbone, spinal column, and spine. To learn more about vertebral column, this site is a very good reference.

CERVICAL SPINE

WHAT IS CERVICAL SPINE?

Cervical spine is the neck area of spine and it consists of seven stacked bones called vertebrae. They are marked C1 to C7 in the above image (Figure A).

WHAT DOES THE CERVICAL SPINE DO?

The Cervical spine provides the following:

- Protecting the spinal cord.
- Supporting the head and allowing movement.
- Providing a safe passageway for vertebral arteries.

CERVICAL SPINE FRACTURE

A fracture to the bones of your spine can result from compression (often from minor trauma in a person with osteoporosis) or be a burst fracture (vertebra that is crushed in all directions) or a fracture-dislocation (mostly from vehicle accidents or falls from heights).

TESTS AND IMAGING

Computed tomography (CT) scan uses X-rays and computers to produce images that are very thin “slices” of the area under examination. A CT scan can show the shape and size of spinal canal, its contents, and the bone around it. It helps diagnose bone spurs, osteophytes, bone fusion and bone destruction from infection or tumor. With the test data, we used it to evaluate the learned model’s generalizability or prediction ability in parallel.

EXPLORATORY DATA ANALYSIS

WORKING WITH KAGGLE DATASET

The Kaggle dataset provides a total of 2018 study instances and they are leveraged as part of the exploratory data analysis in this document. The training images, test images, associated data files, images and NIfTI segment files are available from Kaggle to further explore. We used the training data to help fuel our learning algorithms. With that we were able to produce learned models and patterns. The data was set up within a matrix to understand what was happening pixel by pixel to identify which vector element might be spatially adjacent to which vector element in an image after watching many images and assign a correlation value.

```
paths = {
    'train_df': Path('../input/rsna-2022-cervical-spine-fracture-detection/train.csv'),
    'train_bbox': Path('../input/rsna-2022-cervical-spine-fracture-detection/train_bounding_boxes.csv'),
    'train_images': Path('../input/rsna-2022-cervical-spine-fracture-detection/train_images'),
    'train_nifti_segments': Path('../input/rsna-2022-cervical-spine-fracture-detection/segmentations'),
    'test_df': Path('../input/rsna-2022-cervical-spine-fracture-detection/test.csv'),
    'test_images': Path('../input/rsna-2022-cervical-spine-fracture-detection/test_images')
}
```

Figure B: Data files, images, and NIfTI format segments

Training data column information is represented below:

- StudyInstanceUID - The study ID. There is one unique study ID for each patient scan.
- patient_overall - One of the target columns. There is overall patient level outcome, i.e., if any of the vertebrae are fractured.
- C[1-7] - The other target columns identifies whether the given vertebrae is fractured. Please refer to the above diagram (Figure A) for the real location of each vertebra in the spine.

```
train_df.head()
```

	StudyInstanceUID	patient_overall	C1	C2	C3	C4	C5	C6	C7
0	1.2.826.0.1.3680043.6200	1	1	1	0	0	0	0	0
1	1.2.826.0.1.3680043.27262	1	0	1	0	0	0	0	0
2	1.2.826.0.1.3680043.21561	1	0	1	0	0	0	0	0
3	1.2.826.0.1.3680043.12351	0	0	0	0	0	0	0	0
4	1.2.826.0.1.3680043.1363	1	0	0	0	0	1	0	0

Figure C: Training dataset

Now, let us add “total_fractures” column to the training dataset and get the total fractures for each study instance.

```
train_df['total_fractures'] = train_df.loc[:,[f"C{i}" for i in range(1,8)]].sum(axis=1)
```

```
train_df.head()
```

	StudyInstanceUID	patient_overall	C1	C2	C3	C4	C5	C6	C7	total_fractures
0	1.2.826.0.1.3680043.6200	1	1	1	0	0	0	0	0	2
1	1.2.826.0.1.3680043.27262	1	0	1	0	0	0	0	0	1
2	1.2.826.0.1.3680043.21561	1	0	1	0	0	0	0	0	1
3	1.2.826.0.1.3680043.12351	0	0	0	0	0	0	0	0	0
4	1.2.826.0.1.3680043.1363	1	0	0	0	0	1	0	0	1

Figure D: Includes total fractures column for training dataset

UNDERSTANDING CT SCANS

A CT scan generates images that can be reformatted in multiple planes. It can even generate three-dimensional images. A more comprehensive understanding of CT scans can be found here.

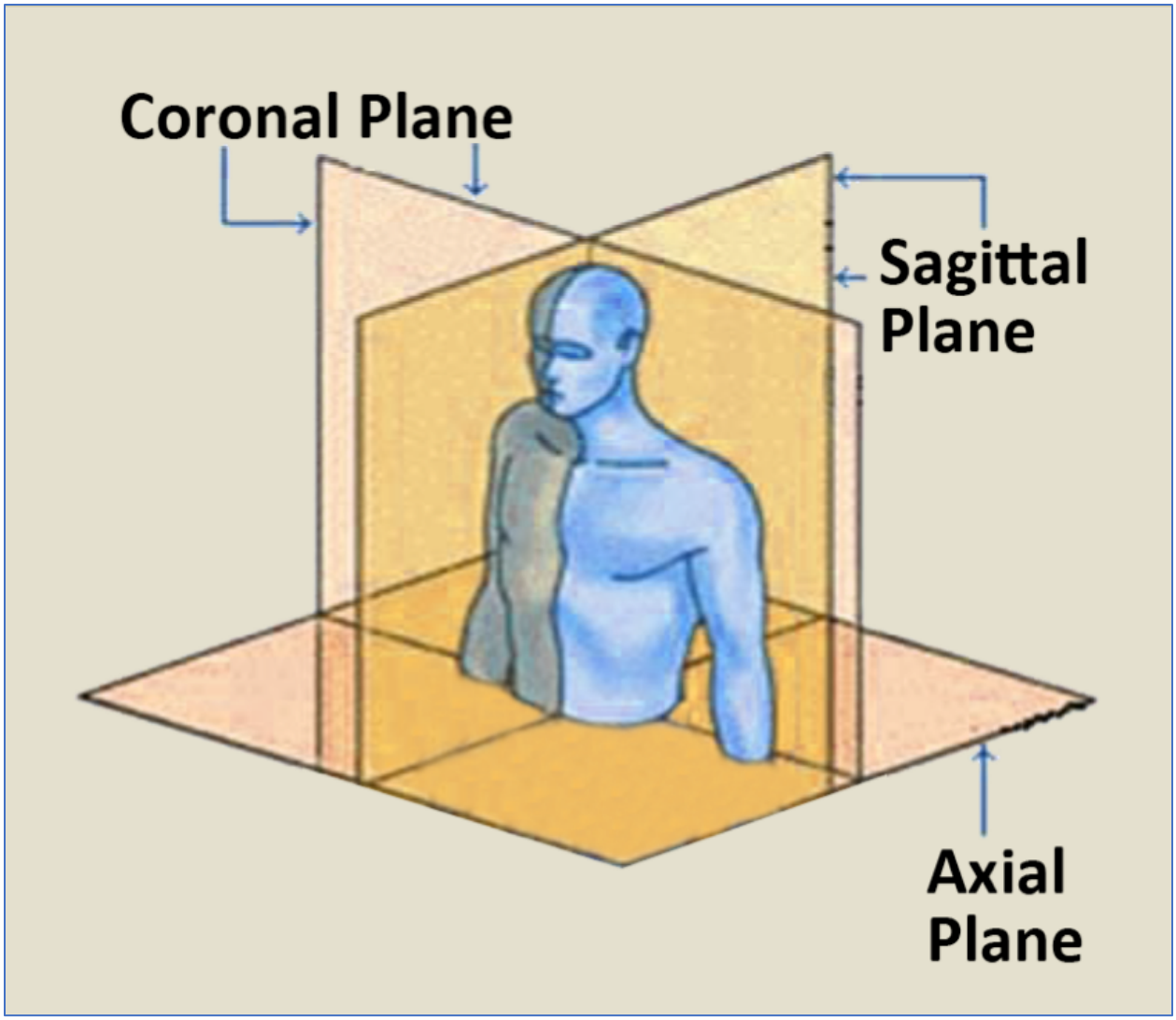


Figure E: CT Scan planes

The data in “train_images” and “test_images” are segmented as per the axial plane (refer to Figure E). Now, let us segmentation paths to each study instance in the dataset.

```
pd.set_option('display.max_colwidth', None)
train_df['segment_path'] = train_df['StudyInstanceUID'].map(lambda x: paths['train_images']/x)
train_df.head()
```

	StudyInstanceUID	patient_overall	C1	C2	C3	C4	C5	C6	C7	total_fractures	segment_path
0	1.2.826.0.1.3680043.6200	1	1	1	0	0	0	0	0	2	../input/rsna-2022-cervical-spine-fracture-detection/train_images/1.2.826.0.1.3680043.6200
1	1.2.826.0.1.3680043.27262	1	0	1	0	0	0	0	0	1	../input/rsna-2022-cervical-spine-fracture-detection/train_images/1.2.826.0.1.3680043.27262
2	1.2.826.0.1.3680043.21561	1	0	1	0	0	0	0	0	1	../input/rsna-2022-cervical-spine-fracture-detection/train_images/1.2.826.0.1.3680043.21561
3	1.2.826.0.1.3680043.12351	0	0	0	0	0	0	0	0	0	../input/rsna-2022-cervical-spine-fracture-detection/train_images/1.2.826.0.1.3680043.12351
4	1.2.826.0.1.3680043.1363	1	0	0	0	0	1	0	0	1	../input/rsna-2022-cervical-spine-fracture-detection/train_images/1.2.826.0.1.3680043.1363

Figure F: Segmentation paths included for each study instance in the dataset

Additionally, each segment has different number of slices associated with study instance. Let us count the number of slices associated with each study instance and include that in the training dataset.

```
def num_slices(path):
    slices = list(path.glob('*'))
    return len(slices)

train_df['num_slices'] = train_df['segment_path'].progress_map(num_slices)
train_df['num_slices'] = train_df['num_slices'].astype('int')
```

100% 2018/2018 [04:38<00:00, 5.29it/s]

```
train_df.head()
```

	StudyInstanceUID	patient_overall	C1	C2	C3	C4	C5	C6	C7	total_fractures	segment_path	num_slices
0	1.2.826.0.1.3680043.6200	1	1	1	0	0	0	0	0	2	../input/rsna-2022-cervical-spine-fracture-detection/train_images/1.2.826.0.1.3680043.6200	243
1	1.2.826.0.1.3680043.27262	1	0	1	0	0	0	0	0	1	../input/rsna-2022-cervical-spine-fracture-detection/train_images/1.2.826.0.1.3680043.27262	406
2	1.2.826.0.1.3680043.21561	1	0	1	0	0	0	0	0	1	../input/rsna-2022-cervical-spine-fracture-detection/train_images/1.2.826.0.1.3680043.21561	385
3	1.2.826.0.1.3680043.12351	0	0	0	0	0	0	0	0	0	../input/rsna-2022-cervical-spine-fracture-detection/train_images/1.2.826.0.1.3680043.12351	501
4	1.2.826.0.1.3680043.1363	1	0	0	0	0	1	0	0	1	../input/rsna-2022-cervical-spine-fracture-detection/train_images/1.2.826.0.1.3680043.1363	199

Figure F: Number of segment slices computed and included in the training dataset

Furthermore, let us review the training dataset class imbalance and if any exists. As observed in Figure G, it is clear that there is very good class distribution exists in the training dataset.

```
fig = px.pie(train_df['patient_overall'].value_counts().reset_index(),
       names=['not_fractured','fractured'], # [0,1]
       values='patient_overall',
       color_discrete_sequence = px.colors.qualitative.Pastel,
       title='Patient Fracture Overall Distribution'
      ).update_traces(textinfo='label+percent')
iplot(fig)
```

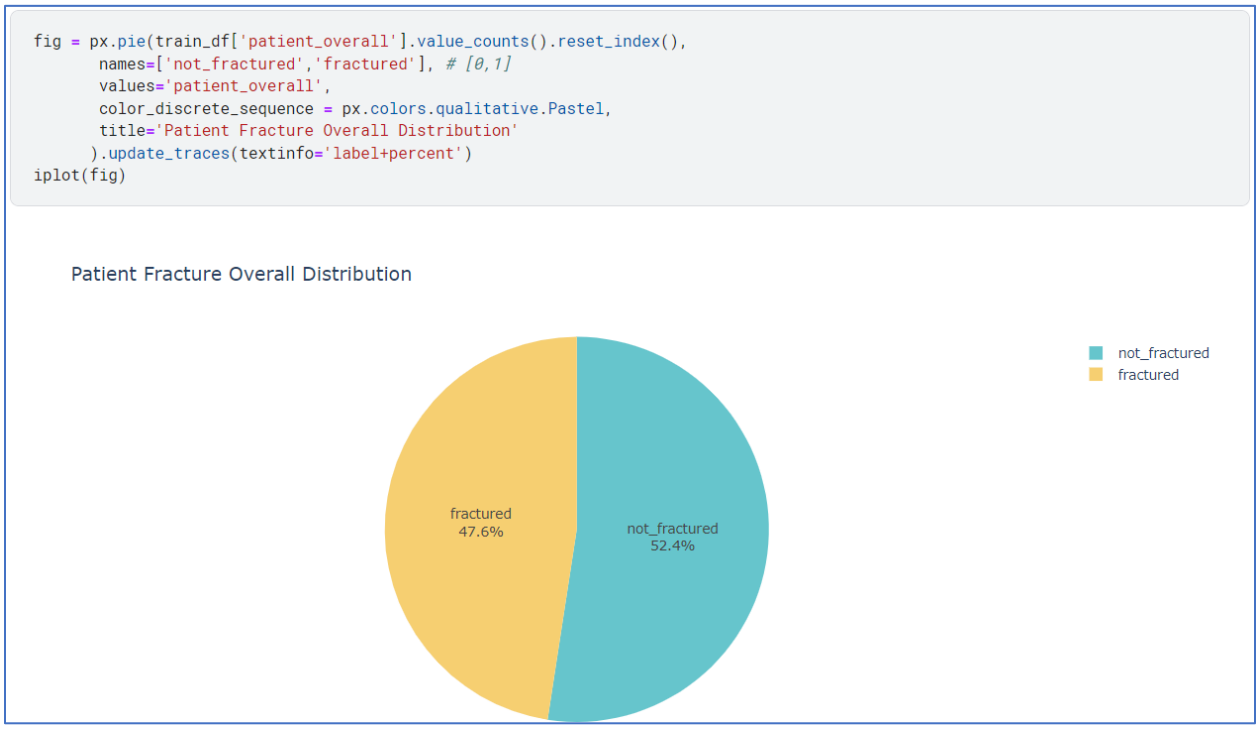


Figure G: Class distribution

While there is good class distribution exists from overall fracture perspective, we can't say the about fractures per vertebra as seen in Figure H for all 2018 study instance.

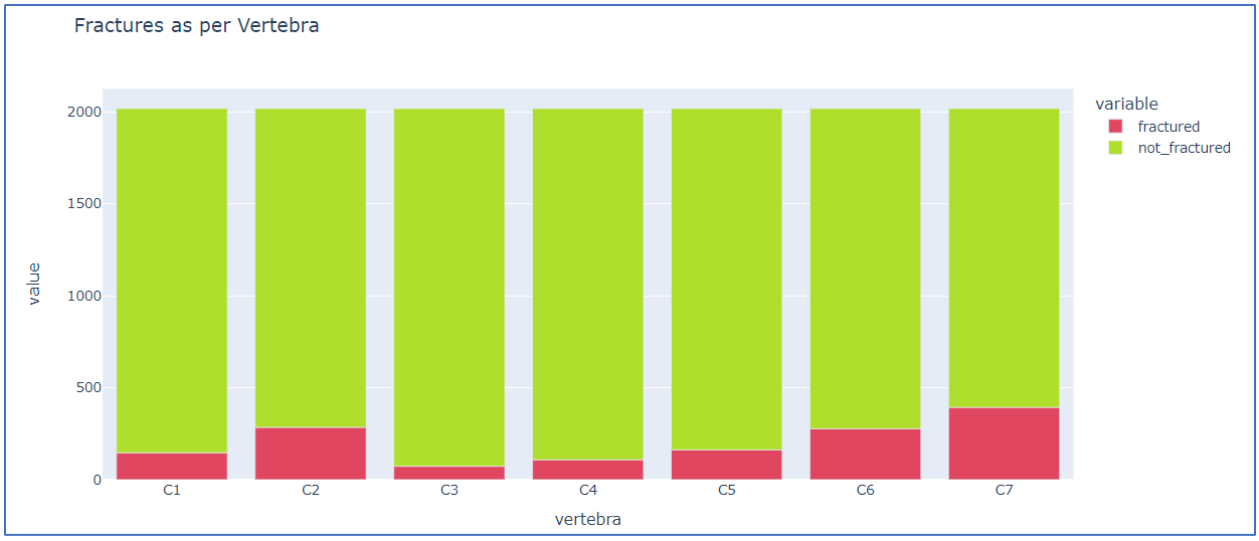


Figure H: Fractures per vertebra

Now, just to get a further idea on total fractures across all study instances for each vertebra, below distribution (Figure I) provides C1-C7 fracture counts.

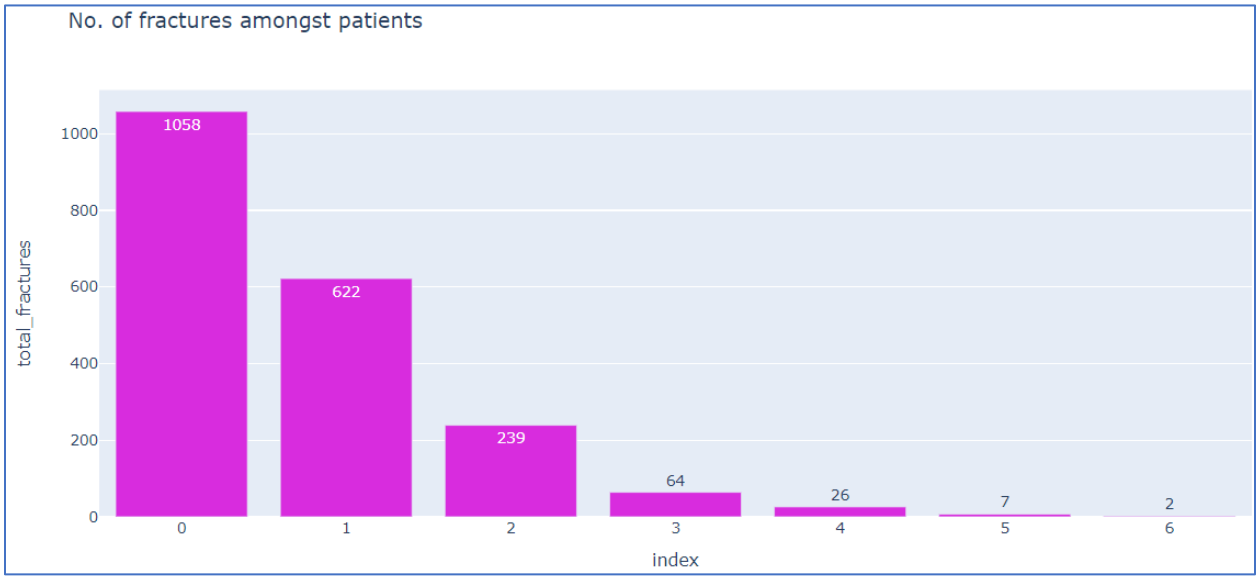


Figure I: Number of fractures for each vertebra across all study instances

Further, to round up the training dataset analysis, let us review the number of slices distribution histogram across all study instances (Figure J).

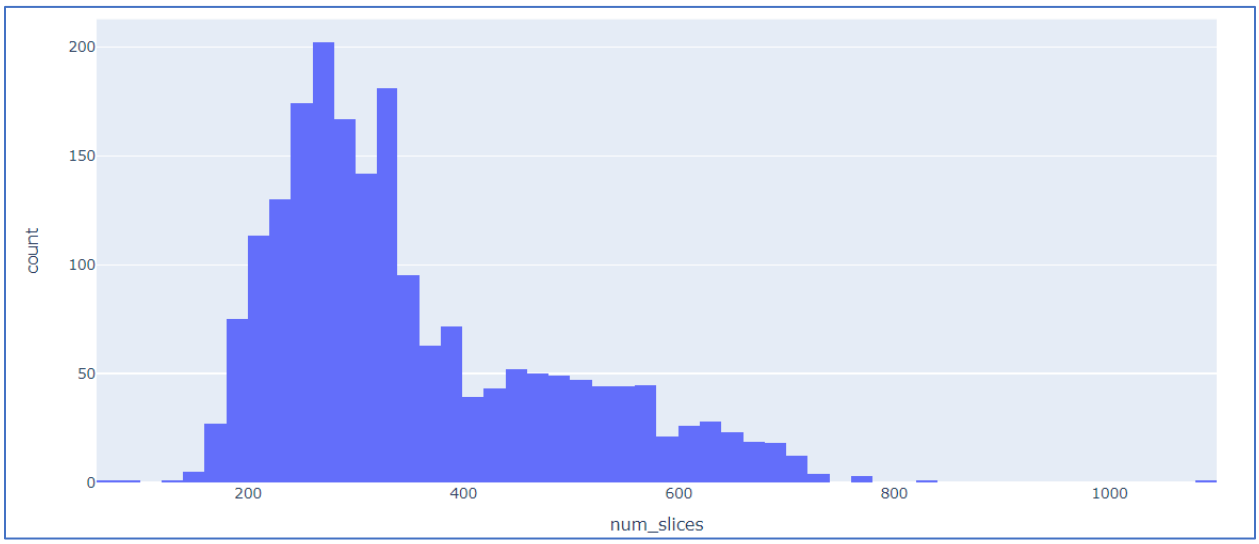


Figure K: Number of slices distribution histogram

DCM AND NII IMAGE FILE FORMATS REVIEW

- .dcm - DICOM file: A DCM file is an image file saved in the Digital Imaging and Communications in Medicine (DICOM) image format. It stores a medical image, such as a CT scan or ultrasound, and may also include patient information to pair the image with the patient.

- .nii - NIFTI file format - A nibabel image object is the association of three things:

    - an N-D array containing the image data.
    - a (4, 4) affine matrix mapping array coordinates to coordinates in some RAS+ world coordinate space (Coordinate systems and affines).

- image metadata in the form of a header.

DCM IMAGE FILE

First let us work with DCM file (Figure L).

```
random_dcm_file = list(train_df['segment_path'][123].glob('*'))[0]
print(random_dcm_file)
random_dcm_file = pydicom.dcmread(random_dcm_file)
print(random_dcm_file)
```

```
../input/rsna-2022-cervical-spine-fracture-detection/train_images/1.2.826.0.1.3680043.21724/257.dcm
Dataset.file_meta -------------------------------
(0002, 0001) File Meta Information Version       OB: b'\x00\x01'
(0002, 0002) Media Storage SOP Class UID         UI: CT Image Storage
(0002, 0003) Media Storage SOP Instance UID      UI: 1.2.826.0.1.3680043.21724.1.257
(0002, 0010) Transfer Syntax UID                 UI: Implicit VR Little Endian
(0002, 0012) Implementation Class UID            UI: 1.2.40.0.13.1.1.1
(0002, 0013) Implementation Version Name         SH: 'PYDICOM 2.3.0'
-------------------------------------------------
(0008, 0018) SOP Instance UID                    UI: 1.2.826.0.1.3680043.21724.1.257
(0008, 0023) Content Date                        DA: '20220727'
(0008, 0033) Content Time                        TM: '181031.112496'
(0010, 0010) Patient's Name                      PN: '21724'
(0010, 0020) Patient ID                          LO: '21724'
(0018, 0050) Slice Thickness                     DS: '0.625'
(0020, 000d) Study Instance UID                  UI: 1.2.826.0.1.3680043.21724
(0020, 000e) Series Instance UID                 UI: 1.2.826.0.1.3680043.21724.1
(0020, 0013) Instance Number                     IS: '257'
(0020, 0032) Image Position (Patient)            DS: [-87.257, -14.763, -149.388]
(0020, 0037) Image Orientation (Patient)         DS: [1.000000, 0.000000, 0.000000, 0.000000, 1.000000, 0.000000]
(0028, 0002) Samples per Pixel                   US: 1
(0028, 0004) Photometric Interpretation          CS: 'MONOCHROME2'
(0028, 0010) Rows                                US: 512
(0028, 0011) Columns                             US: 512
(0028, 0030) Pixel Spacing                       DS: [0.296875, 0.296875]
(0028, 0100) Bits Allocated                      US: 16
(0028, 0101) Bits Stored                         US: 16
(0028, 0102) High Bit                            US: 15
(0028, 0103) Pixel Representation                US: 1
(0028, 1050) Window Center                       DS: '500.0'
(0028, 1051) Window Width                        DS: '2000.0'
(0028, 1052) Rescale Intercept                   DS: '-1024.0'
(0028, 1053) Rescale Slope                       DS: '1.0'
(7fe0, 0010) Pixel Data                          OW: Array of 524288 elements
```

Figure L: Load a random DCM file for display

Let us review the DCM image file metadata (Figure M).

```
# getting metadata with .get(key)
print("Instance Number:",random_dcm_file.get('InstanceNumber'))
print("Rows x Columns:", random_dcm_file.get("Rows"), random_dcm_file.get("Columns"))
print("Image Position (Patient):", random_dcm_file.get("ImagePositionPatient"))
```

```
Instance Number: 257
Rows x Columns: 512 512
Image Position (Patient): [-87.257, -14.763, -149.388]
```

Figure M: DCM file metadata attributes extraction

From the above metadata (Figure L and Figure M):

- Patient ID = Patient Name
- Slice Thickness = 1.0 mm [might not be same for all patients]
- Instance Number = slice number
- Image Position (Patient): gives an array of values [x-axis, y-axis, z-axis], here z-axis denotes the position in the sagittal plane

apply_voi_lut() applies a windowing function to the image - read more here

```
dcm_image = apply_voi_lut(random_dcm_file.pixel_array, random_dcm_file)
```

```
plt.imshow(dcm_image, cmap='bone')
```

```
<matplotlib.image.AxesImage at 0x7fa2b8a51c90>
```

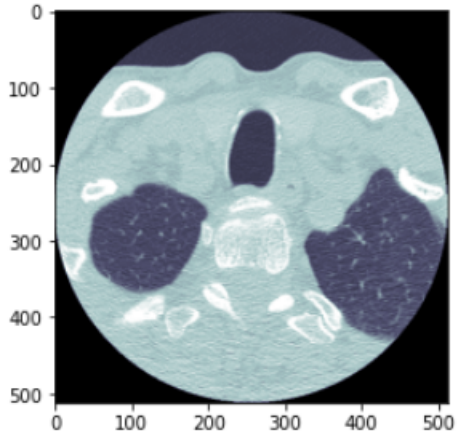


Figure N: DCM Image view

NII IMAGE FILE

From review the data files, not all study instances have NII image files. In fact, there are less than 87 NII image files available out of 2018 study instances. Let us randomly review on the file (Figure O).

```
random_nii_file = list(paths['train_nifti_segments'].glob('*'))[10]
print(random_nii_file)

def open_nii_file(path):
    f = nib.load(path)
    segmentations = f.get_fdata()[:, ::-1, ::-1].transpose(2, 1, 0)
    return segmentations

nii_segments = open_nii_file(random_nii_file)
print(nii_segments.shape, "=> (num_slices, height, width)")
plt.imshow(nii_segments[123,:,:],cmap='bone')
```

```
../input/rsna-2022-cervical-spine-fracture-detection/segmentations/1.2.826.0.1.3680043.32436.nii
(271, 512, 512) => (num_slices, height, width)
<matplotlib.image.AxesImage at 0x7fa2b8049990>
```

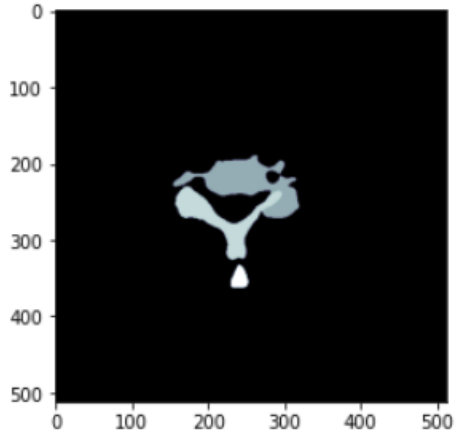


Figure O: NII Image file view

A caveat of axial plane is that we are not aware of which cervical vertebra we are looking at, the above NII segments array can be reused to get the vertebra index using np.unique() – NumPy function (Figure P).

```
print("[background, *, vertebra]")
print("here 4. denotes that the slice is a part of C4 vertebra")
np.unique(nii_segments[100])
```

```
[background, *, vertebra]
here 4. denotes that the slice is a part of C4 vertebra
array([0., 3., 4.])
```

Figure P: Identify the vertebra slice

Now we know that there are only 87 NII files available as noted from Figure Q.

```
f"We only have {len(list(paths['train_nifti_segments'].glob('*')))} nii files / {len(train_df)} studies"
```

```
'We only have 87 nii files / 2018 studies'
```

Figure Q: Count the number of NII files available

Let us add the NII segments path to the training dataset (Figure R).

```
def add_nii_segment_path(uid):
    base_path = paths['train_nifti_segments']
    # path if exists else None
    path = base_path/(uid+'.nii')
    if path.exists():
        return path
    return None

train_df['nii_segments_path'] = train_df['StudyInstanceUID'].map(add_nii_segment_path)
train_df.head()
```

	StudyInstanceUID	patient_overall	C1	C2	C3	C4	C5	C6	C7	total_fractures	segment_path	num_slices	nii_segments_path
0	1.2.826.0.1.3680043.6200	1	1	1	0	0	0	0	0	2	../input/rsna-2022-cervical-spine-fracture-detection/train_images/1.2.826.0.1.3680043.6200	243	None
1	1.2.826.0.1.3680043.27262	1	0	1	0	0	0	0	0	1	../input/rsna-2022-cervical-spine-fracture-detection/train_images/1.2.826.0.1.3680043.27262	406	None
2	1.2.826.0.1.3680043.21561	1	0	1	0	0	0	0	0	1	../input/rsna-2022-cervical-spine-fracture-detection/train_images/1.2.826.0.1.3680043.21561	385	None
3	1.2.826.0.1.3680043.12351	0	0	0	0	0	0	0	0	0	../input/rsna-2022-cervical-spine-fracture-detection/train_images/1.2.826.0.1.3680043.12351	501	None
4	1.2.826.0.1.3680043.1363	1	0	0	0	0	1	0	0	1	../input/rsna-2022-cervical-spine-fracture-detection/train_images/1.2.826.0.1.3680043.1363	199	../input/rsna-2022-cervical-spine-fracture-detection/segmentations/1.2.826.0.1.3680043.1363.nii

Figure R: NII segment path included in the training dataset

Let us define two functions to get DCM image and NII segment from a given study instance path (Figure S).

```
def get_dcm_images(path):
    paths = list(path.glob('*'))
    paths.sort(key=lambda x:int(x.stem)) # sort based on slice index which is the filename: index.dcm
    data = [pydicom.dcmread(f) for f in paths]
    images = [apply_voi_lut(dcm.pixel_array, dcm) for dcm in data]
    return images

def get_nii_segments(path):
    f = nib.load(path)
    segmentations = f.get_fdata()[:, ::-1, ::-1].transpose(2, 1, 0)
    return segmentations
```

Figure S: get dcm image and get nii segment functions

Let us get a sample index DCM image and NII segment for a given study instance (Figure T).

```
sample_idx = 99
dcm_images = get_dcm_images(train_df['segment_path'][sample_idx])
nii_segments = get_nii_segments(train_df['nii_segments_path'][sample_idx])
print((len(dcm_images), *dcm_images[0].shape), nii_segments.shape)
```

```
(271, 512, 512) (271, 512, 512)
```

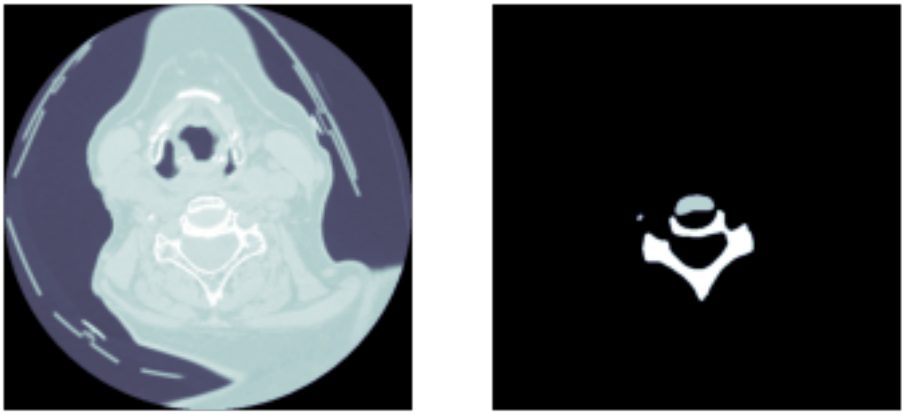
Figure R: Retrieve sample index 99 DCM image and NII segment

Let us plot DCM image and NII segment.

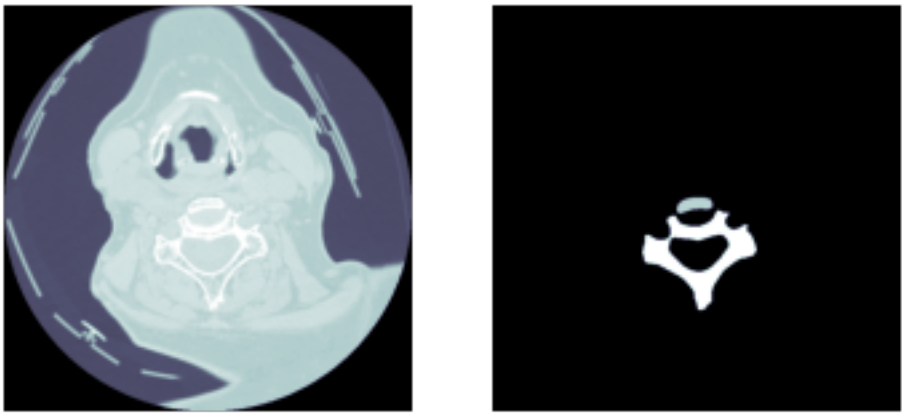
```python
def plot_slice(idx, dcm_images=dcm_images, nii_segments=nii_segments):
    fig, (ax1, ax2) = plt.subplots(1, 2)
    ax1.axis('off'); ax2.axis('off')
    fig.suptitle(f'Slice {idx}')
    ax1.imshow(dcm_images[idx], cmap='bone')
    ax2.imshow(nii_segments[idx,:,:], cmap='bone')

for i in range(123,128):
    plot_slice(i)
```

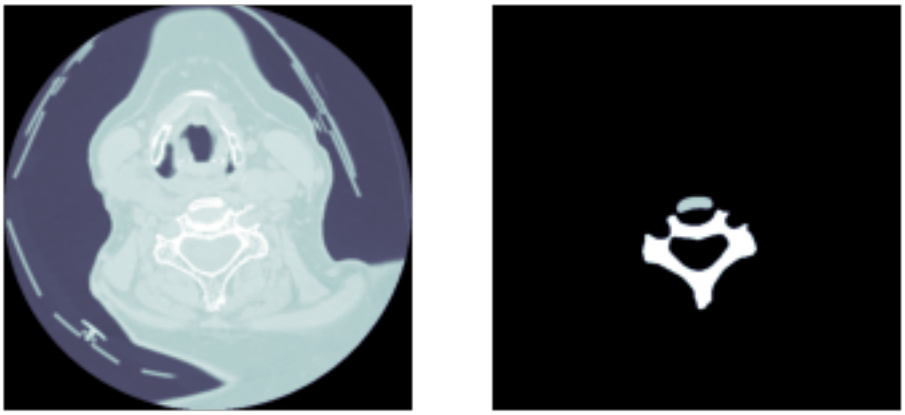
Slice 123



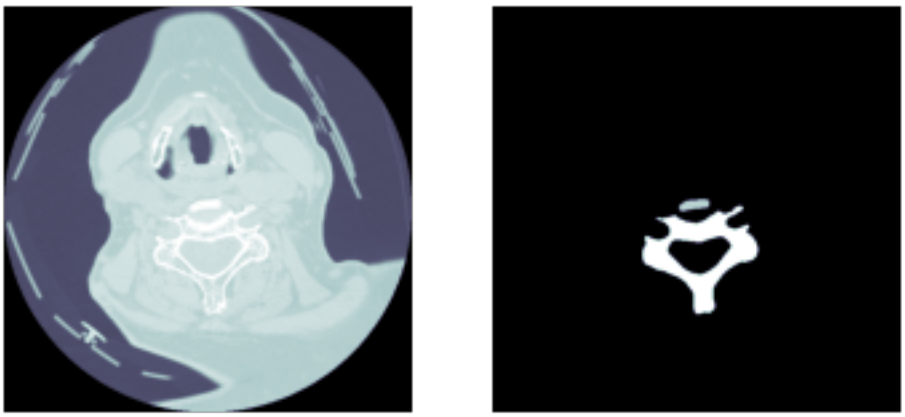
Slice 124



Slice 125



Slice 126



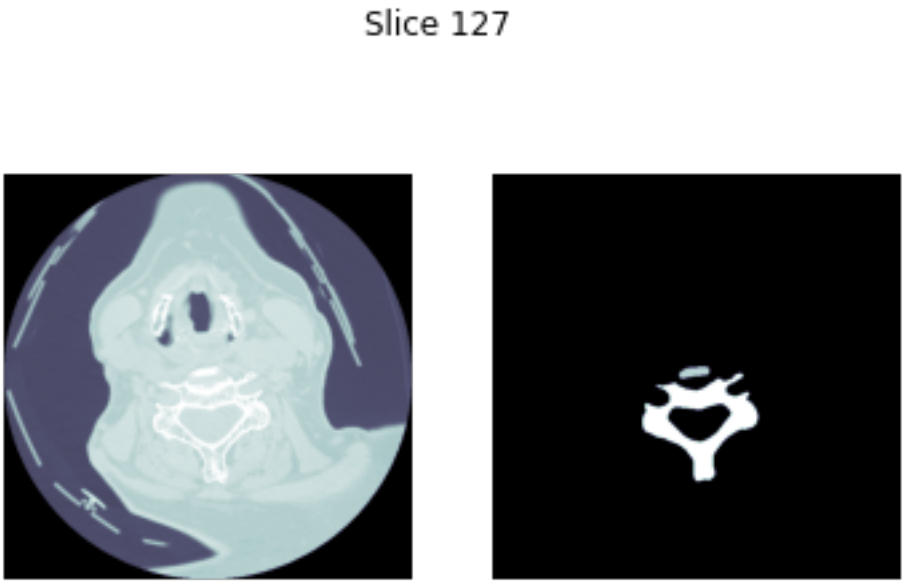


Figure S: DCM Images and NII Segments associated with given slice