**Udacity: AI for Healthcare Nanodegree (May 2025)**

**Course Summary:**

The "AI for Healthcare" Nanodegree provides a comprehensive, project-based curriculum that integrates core concepts of medical imaging, deep learning, and clinical deployment of AI systems. Beginning with foundational knowledge in 2D medical imaging, students learn how to work with real-world clinical data formats such as DICOM, perform exploratory data analysis (EDA), and build convolutional neural networks (CNNs) to classify pathologies like pneumonia from chest X-rays. Emphasis is placed on understanding image metadata, demographic balancing, and pre-processing techniques such as normalization and augmentation tailored to clinical imaging. Learners are trained to split datasets intelligently, taking into account positive/negative class distributions and real-world imbalances, and to evaluate models using appropriate performance metrics (e.g., F1 score, recall, precision) rather than simple accuracy. They develop end-to-end pipelines from DICOM ingestion to model output and implement wrappers that align with clinical workflow integration. Regulatory readiness is a key focus, culminating in the preparation of a mock FDA 510(k) submission, including intended use statements, model limitations, silver-standard labeling approaches, and performance validation planningAI\_for\_HealthCare\_NanoD….

The course then expands into 3D medical imaging, where students explore data formats like NIFTI alongside DICOM, and learn advanced processing techniques such as multi-planar reconstruction, 3D segmentation with U-Net, and registration. Imaging modalities covered include CT and MRI, with technical deep dives into resolution, windowing, and the use of contrast agents. Clinical use cases are framed as machine learning problems, emphasizing prioritization of workflow enhancement over mere diagnosis. Learners practice selecting appropriate modeling strategies—such as 2D, 2.5D, or 3D convolutional architectures—based on task complexity and computational efficiency. Evaluation metrics for segmentation (e.g., Dice coefficient, Hausdorff distance) are introduced, and practical knowledge of voxel geometry and coordinate systems is reinforcedAI\_for\_HealthCare\_NanoD….

Throughout, the course blends technical skill-building with system-level thinking, including how to frame problems, curate datasets, validate performance, and ensure clinical relevance. Tools used include PyTorch, Keras, and 3D Slicer, and projects are executed within GPU-powered Jupyter environments to simulate real-world constraints. Regulatory and ethical considerations are woven throughout, including the importance of specificity and sensitivity tradeoffs, FDA classification pathways (Class I–III), and post-deployment device monitoring. This course ultimately empowers learners not only to build clinically robust models, but also to think like AI engineers operating at the intersection of healthcare, data science, and regulation.

**Course 1+2: Applying AI to 2d Medical Imaging Data**

**Part 1: Introduction to AI for 2D Medical Imaging**

In this project, you will

1. Distill data that are useful for training algorithms to detect pneumonia from a giant set of chest x-ray images taken from actual patients.

2. Build a CNN model to detect the presence or absence of pneumonia.

3. Build wrappers that read medical images from their real-world clinical formats (DICOM). 4. Write up a documentation & validation plan of your algorithm for FDA 510(k) submission.

**Part 2: Clinical Foundations of 2D Medical Imaging**

Exercise: 2D imaging algorithm:

Though this is not typically thought of as 2D imaging one could consider EEG readings as a potential 2-D image that could be integrated into an algorithm as part of medical imaging. For example, one could envision a 2-D image from EEG that could classify whether a patient has generalized anxiety disorder, or is normal. This algorithm could be integrated as part of a workflow in an outpatient setting, where patients are being diagnosed, perhaps in a mood disorder or anxiety disorder clinic. The FDA would likely classify the algorithm as similar to others and probably class one since the algorithm itself would be what is being implemented. The EEG is already a known medical device that has been approved and this algorithm would simply take the EEG to the image and classify patients as either having generalized anxiety disorder or not.

**Part 3: 2D Medical Imaging Exploratory Data Analysis**

DICOM: Digital Imaging and Communications in Medicine

- acquisition method, images, patient information are also contained in the DICOM file

- the acquisition info, imaging data, and patient information (from tech) is compiled into final DICOM file.

- DICOM was developed in 1993 by Radiology for the purposes of interoperability at multiple hospitals

- One of the most successful implementations of an interoperable standard in healthcare.

DICOM: series & studies

- single 2D series (i.e. single X-ray)

- collection of 2D series (i.e. multiple X-rays)

- together is considered a DICOM study

- DICOM header: Patient, Study, Series, Equipment

- components of DICOM file: header (attributes except pixel data: such as patient demographics, Study Date/Time, Referring physician, Series UID, Number, Equipment, Image Number/Type, Acquisition attributes, Position attributes, Resolution), image (pixel data) is not part of DICOM header but is in the DICOM file.

- Patient Health Information (PHI) any ID info, demographics, insurance info, also image file can be identifiable and is considered PHI

- Clinical Data: not part of DICOM (they are in the EMR)

- Radiologist report: included in PACS/EHR, but not part of DICOM; includes location, physician, radiologist, examination, findings, impression

File Types: XML, JSON, CSV, DICOM (Pydicom)

- multiple libraries to work with DICOM, but in python can use pydicom package

DICOM header: what to include in Machine Learning algorithms (pre-screening stage)

- patient ID (do not want same patient in training/testing sets)

- patient sex (equal M/F in training/testing)

- patient age (equal age in both sets)

- do not want to repeat over-training on same study/series

- can use dicom header: ex: screen patient age, screen study for particular image type (ex: one series has the proper body part),

Summary: Reach DICOM, inspect images from DICOM series, inspect header data from single DICOM files

Exploring Population MetaData

- data scientist needs to select relevant subsets of data

- need to merge with patient history

- need image labels (diagnostic labels from radiologist reports or label with radiologist)

- Histograms for single variable distribution (ex: age distribution for patient’s with a particular illness)

- Scatterplots: relationship between two variables (Pearson Correlation Coefficient)

- Co-Occurence Matrix: Heart disease & Obesity (higher value indicates two diseases more likely to co-occur together).

Lesson Conclusion:

DICOM standard, Patient study & series, DICOM attributes and image components

Image attributes (read & extract via pydicom)

Prepare non-image data from DICOM header

Explore Metadata from Populations (Data features for training including histograms, scatterplots, co-occurence matrices can be inputs in machine learning algorithms).

**Part 4: Classification of 2D Medical Images**

- In this lesson, will focus on deep learning (CNN architecture for classification), pre-process data, train/test for different clinical applications

- Outline: types of models, dataset splitting, gold standards (label), image pre-processing for deep learning, CNN Fine tuning for medical tasks, model evaluation (classification)

- CNN for clinical use (vs. basic images for labeling)

- Deep Learning for Medicine: develop intuition for build, train, fine-tune for medical classification tasks

- Machine Learning: Segmentation: (outlines around specific findings), Localization (id region and draws bounding box), Classification

- Prior to Deep Learning: researchers had to predefined features in images important for segmentation, localization, classification

- Classic Machine Learning: this was considered an art, and took a significant amount of researchers time, would require expert input, features were defined and then provided to an algo (ex: Support Vector Machines, Logistic regression) to determine how these features differentiate between classes of images.

- Example Otsu’s method as classic machine learning example. Finds the intensity threshold within images to minimize variance between classes (i.e. foreground vs. background). This is used for segmentation of breast and background. Can extend Otsu’s method for classification (ex: healthy vs. diseased lung tissue).

- Deep Learning: start with algo architecture (do not need to select features), here the deep learning algo would discover features by looking at many images, and seeing what made certain images different from each other, also returns an output on whether a specific finding is present in an image (i.e. classification).

- Convolutional neural network: architecture was designed to mirror human visual cortex in brain. Cells in V1 (receptive), V2 (larger receptive field), V4, IT. Higher level layers can detect more complex pictures. Each layer is downsampled to see larger field.

-UNET for 3D segmentation

- Splitting your dataset: training vs. validation, medical imaging algo need to curate and organize data,

- split data to training (learn the features for classes), and validation set (user will use to determine whether algo classifies and performance); generally data should be split 80% of positive cases (training) and 20% of positive cases (validation). No image should be used for both datasets.

- Training set: balanced for positive/negative cases, distributed for demographics (same distribution as overall dataset), real-world sets do not include balanced cases, rather real-world balance used.

- Splitting example: 100 cases, 30% pos/ 70% neg. Then split by 80:20 on positive cases for training/validation. Training set is not balanced for positive/negative. Next step is to discard negative cases so that there are even number of pos/neg cases. Keras libraries (train\_test\_split) with test\_size = 0.2, stratify by positive cases. However, validation set can be imbalanced to reflect the real-world situation where negative cases would be more prevalent.

- Gold standard labeling: detects disease with highest sensitivity and specificity (accuracy). In pneumonia, the gold standard is biopsy, thus will rely on radiologist for labeling as pathology will take time. In mammography, the gold standard is radiologist read, but if mass is found need biopsy to determine if malignant / benign.

- Gold standard is often times unattainable for algo developer, only get a large set of DICOM images. Thus, a lot of information is not available (radiologist report, biopsy/ digital pathology). Thus, need to establish a ground truth that is close to the gold standard.

- Ground truth: set of labels that determine which class (ex: biopsy based labeling, NLP-extracted labels from radiologist reports, one or more radiologist interpretation, output of state-of-the-art algorithm). Challenges: biopsy data is expensive / hard to get, NLP can have inaccuracies, limited dataset (ex: only x-rays without labels, need to hire radiologist to go through each image and label them).

- Silver standard (hire several radiologists to provide diagnosis of image, final diagnosis is determined by a voting system including experiences as part of a voting system). This is alternative way to arrive at Ground Truth.

Image Pre-processing for Model Training

- Goals: remove image noise (e.g. background extraction), enforce some normalization across images (zero-mean, standardization), enlarge your dataset (image augmentation), resize for your CNN architecture’s required input.

- Intensity normalization: first remove background pixels (Otsu’s method), then for convolutional neural network architecture perform standardize a normalization by subtracting the mean intensity value and divide by standard deviation. This will result in mean centering around zero and limit range of intensity values to ensure that CNN filters weights do not reach infinity.

- Image augmentation: create different versions of original images to add heterogenous examples to try to mimic real-world variation. \*Keras\* provides a package called “ImageDataGenerator” for image augmentation (added images). Example parameters: horizontal flip (set to true), height\_shift\_range, width\_shift\_range (body shapes), rotation\_range (images that are not fully aligned), shear\_range (how much to pull images), zoom\_range (how much to zoom in parts of image). Need to consider these variables within limits. [\*\*Goal of image augmentation is to give neural network more examples of what images could look like in real world, thus not all types of augmentation are appropriate, example is vertical\_flip which is not reflective of real world scenario]. \*\*Validation Data should never be augmented. Just like we didn’t create an artificial balance of positive and negative cases in our validation set, we should never augment as want it reflect real world. Validation data should still be normalized so that intensity values are close to zero, but other than that do not want to modify images.

Fine-tuning CNNs for 2D medical image classification

- start with pre-trained models (faster)

- CNN: 224 x 224 x 3 -> gets downsampled to different sizes via “max pooling”

- Re-use CNN architecture: first set of layers can be re-used or frozen, if the properties are general, later or downstream layers can be fine-tuned on a different data set with similar goals to allow saving time / energy for novel models

- Ex: Fine-tuning a pre-trained VGG16 model to detect tumors in 2D chest x-rays

- One of the key pieces of fine-tuning is the last layer. We need to adjust the dimension of the last layer to match our specific use cases. We can also add new layers to train from scratch.

Evaluating the Model

- monitor performance of model on testing and training test

- each time entire training data passes through CNN (called Epoch 1)

- at end of each Epoch, CNN has a loss function to determine how different its prediction is from ground truth from training images (training loss).

- Network then uses training loss to update weights of every single filter of each layer that is being trained; to make weights more accurate in next epoch via back propagation

- Need to monitor loss over time to see how much to train

- Also use loss function (validation loss) to see how prediction matches validation classification (we do not update weights at this stage), not teaching how to be a better model; thus do not update weights via back propagation.

- Training Loss and Validation Loss at end of each epoch

- Goal: want to see slowly decreasing loss function for both training and validation samples. Here can save weights when loss function is more stable

- Evaluating epoch after epoch: Use function from “keras" package called fit.generator to obtain loss values after each epoch to plot and determine at which step to freeze weights

- To avoid overfitting (where validation loss stops decreasing after a few epochs), one can change some of the parameters such as “batch size”, “learning rate”, “dropout” and add “more variation to the training data.”

Summary: Types of Models, Dataset Splitting, Gold Standards, Image Pre-processing, CNN layers and filters and Fine Tuning, Model Evaluation

Section 5: Translating AI Algorithms for Clinical Settings with the FDA

- FDA regulatory process (intended use) -> Algo limitations -> Performance Statistics (for device) -> FDA validation plan

- Even best CNN with great performance on benchmark datasets still will need to fit real-world clinical settings to help clinicians treat real patients with real diseases

- need to understand the nuances of the FDA 510(k) process (take several multi-day courses)

- FDA: intended use statement (tells FDA exactly what algorithm is used, ex: ID if cancer is malignant or begin, will be used by FDA to define risk/class of algo). Indications for use statement: how algo \*could\* be used

- Medical device

- Class I low risk (Exempt, 501k): 47% of approved devices on market (minimal contact with patients and low impact on health of patient)

- Class II medium risk (501k, Exempt): general controls are insufficient to provide a reasonable assurance of the safety and effectiveness of device; needs to show that device is substantially equivalent to predicate device (similar use/design/standards/etc). Most AI for medical imaging algo end up going through Class II medical device evaluations (i.e. diagnostic tools)

- Class III high risk (PMA, 501k): only 10% of devices (ex: permanent implants, life support systems, permanent implants). A device that may have initially been put into class II may be bumped up to class III if the manufacturer cannot demonstrate substantial equivalence to a predicate during the FDA process.

- Intended Use statement should be tied to risk and class (ex: scalpel could be Class I or Class III based on this statement)

- If algo is Class III may need PMA (premarket approval) which takes much longer unless can make case for predicate device (i.e. 501K process instead)

- CADx (computer-assisted diagnosis) has recently been classified by FDA as Class 2. However, without a predicate device, algorithm would still have to go through PMA pathway.

- Indications for Use: precise situations and reasons where and why you would use this device. This statement can be “indicated for use in screening mammography studies, for ages between 20-59 age range, without prior history of breast cancer, etc).

Medical Device Reporting:

- After algo is cleared by the FDA and released out into the clinical world, the FDA has a system called medical device reporting to continuously monitor whether or not your algorithm is malfunctioning in the world. Anytime one of the end-users discovers a malfunction in the software, they report back to the manufacturer (i.e. industry/academia) and the manufacturer is required to report it back to the FDA. FDA can recall device or update labeling to acknowledge limitations.

Computational Limitations

- ex: CNN for brain bleeds -> indications for use: to help prioritize radiologist workflow (i.e. assist radiologist)

- indications for use: workflow reprioritization in emergency settings

- requirement: run fast; computational limitations include GPU/cloud usage (does not achieve fast performance without this type of infrastructure)

- Choosing a narrow scope with device is a good strategy especially if can maximize market impact (i.e. marketable algorithm)

- Can always expand your devices functionally over time and work with the FDA to get novel expansions approved.

Translating Performance into Clinical Utility

- accuracy data for algorithm is not sufficient as disease labels are low probability

- negative cases can be assessed with specificity (TN/ (TN + FP)), proportion of accurately-identified negative cases

- specificity will be high even if algo says all negative cases are negative

- sensitivity is more frequently used a clinical performance metric: TP / (TP + FN), proportion of accurately-identified positive cases

- specificity ignores true positives and sensitivity does not account for false positives.

- precision: tp / (tp + fp) [positive predicted value]: high precision test gives more confidence that positive test is actually positive; could still miss positive cases

- recall: allow us to confidently rule out disease; gives confidence that negative test is truly negative

- optimizing one metric comes at expense of the other (there is a balance)

Performance Trade-offs

- varying classification threshold is example of where tradeoff in diagnostic properties would be present

- output of CNN is not usually 0 or 1 but is a threshold score (probability between 0 to 1)

- recall and precision values will vary when different threshold values used; can generate a precision-recall curve

- visualize a curve

- for binary classification problems, there’s a score called the F1 score which combines both precision and recall and allows us to better measure a test accuracy when there are class imbalances. Harmonic mean of precision and recall. This can be a single score or metric.

Designing an FDA validation plan

- FDA validation dataset, then ground truth labels (radiologist or group), performance standard from literature, run algo on data to meet performance standard, then submit to FDA.

Final Exercise:

- need validation dataset that is screening mammography between ages 40-80

- for Algo A: no implants, distribution of densities reflects real world, silver standard approach using mixture of radiologists

- for Algo B: no prior hx of breast cancer, single radiologist label as they are really accurate at this task.

- Summary: FDA risk categories, algo limitations using medical device reporting, precision/recall tradeoff, validation plan for FDA

Final Project (Detecting Pneumonia Detection from Chest X-rays)

Pneumonia Detection from Chest X-Rays

Project Overview

In this project, you will apply the skills that you have acquired in this 2D medical imaging course to analyze data from the NIH Chest X-ray Dataset and train a CNN to classify a given chest x-ray for the presence or absence of pneumonia. This project will culminate in a model that can predict the presence of pneumonia with human radiologist-level accuracy that can be prepared for submission to the FDA for 510(k) clearance as software as a medical device. As part of the submission preparation, you will formally describe your model, the data that it was trained on, and a validation plan that meets FDA needs.

You will be provided with the medical images with clinical labels for each image that were extracted from their accompanying radiology reports.

The project will include access to a GPU for fast training of deep learning architecture, as well as access to 112,000 chest x-rays with disease labels acquired from 30,000 patients.

Pneumonia and X-Rays in the World

Chest X-ray exams are one of the most frequent and cost-effective types of medical imaging examinations. Deriving clinical diagnoses from chest X-rays can be challenging, however, even by skilled radiologists.

When it comes to pneumonia, chest X-rays are the best available method for diagnosis. More than 1 million adults are hospitalized with pneumonia and around 50,000 die from the disease every year in the US alone. The high prevalence of pneumonia makes it a good candidate for the development of a deep learning application for two reasons: 1) Data availability in a high enough quantity for training deep learning models for image classification 2) Opportunity for clinical aid by providing higher accuracy image reads of a difficult-to-diagnose disease and/or reduce clinical burnout by performing automated reads of very common scans.

The diagnosis of pneumonia from chest X-rays is difficult for several reasons:

The appearance of pneumonia in a chest X-ray can be very vague depending on the stage of the infection

Pneumonia often overlaps with other diagnoses

Pneumonia can mimic benign abnormalities

For these reasons, common methods of diagnostic validation performed in the clinical setting are to obtain sputum cultures to test for the presence of bacteria or viral bodies that cause pneumonia, reading the patient's clinical history and taking their demographic profile into account, and comparing a current image to prior chest X-rays for the same patient if they are available.

About the Dataset

The dataset provided to you for this project was curated by the NIH specifically to address the problem of a lack of large x-ray datasets with ground truth labels to be used in the creation of disease detection algorithms.

The data is mounted in the Udacity Jupyter GPU workspace provided to you, along with code to load the data. Alternatively, you can download the data from the kaggle website or official NIH website and run it locally. You are STRONGLY recommended to complete the project using the Udacity workspace since the data is huge, and you will need GPU to accelerate the training process.

There are 112,120 X-ray images with disease labels from 30,805 unique patients in this dataset. The disease labels were created using Natural Language Processing (NLP) to mine the associated radiological reports. The labels include 14 common thoracic pathologies:

Atelectasis

Consolidation

Infiltration

Pneumothorax

Edema

Emphysema

Fibrosis

Effusion

Pneumonia

Pleural thickening

Cardiomegaly

Nodule

Mass

Hernia

The biggest limitation of this dataset is that image labels were NLP-extracted so there could be some erroneous labels but the NLP labeling accuracy is estimated to be >90%.

The original radiology reports are not publicly available but you can find more details on the labeling process here.

Dataset Contents:

112,120 frontal-view chest X-ray PNG images in 1024\*1024 resolution (under images folder)

Meta data for all images (Data\_Entry\_2017.csv): Image Index, Finding Labels, Follow-up #, Patient ID, Patient Age, Patient Gender, View Position, Original Image Size and Original Image Pixel Spacing.

Project Steps

1. Exploratory Data Analysis (completed)

The first part of this project will involve exploratory data analysis (EDA) to understand and describe the content and nature of the data.

Note that much of the work performed during your EDA will enable the completion of the final component of this project which is focused on documentation of your algorithm for the FDA. This is described in a later section, but some important things to focus on during your EDA may be:

The patient demographic data such as gender, age, patient position,etc. (as it is available)

The x-ray views taken (i.e. view position)

The number of cases including:

number of pneumonia cases,

number of non-pneumonia cases

The distribution of other diseases that are comorbid with pneumonia

Number of disease per patient

Pixel-level assessments of the imaging data for healthy & disease states of interest (e.g. histograms of intensity values) and compare distributions across diseases.

2. Building and Training Your Model

Training and validating Datasets

From your findings in the EDA component of this project, curate the appropriate training and validation sets for classifying pneumonia. Be sure to take the following into consideration:

Distribution of diseases other than pneumonia that are present in both datasets

Demographic information, image view positions, and number of images per patient in each set

Distribution of pneumonia-positive and pneumonia-negative cases in each dataset

Model Architecture

In this project, you will fine-tune an existing CNN architecture to classify x-rays images for the presence of pneumonia. There is no required architecture required for this project, but a reasonable choice would be using the VGG16 architecture with weights trained on the ImageNet dataset. Fine-tuning can be performed by freezing your chosen pre-built network and adding several new layers to the end to train, or by doing this in combination with selectively freezing and training some layers of the pre-trained network.

Image Pre-Processing and Augmentation

You may choose or need to do some amount of preprocessing prior to feeding imagees into your network for training and validating. This may serve the purpose of conforming to your model's architecture and/or for the purposes of augmenting your training dataset for increasing your model performance. When performing image augmentation, be sure to think about augmentation parameters that reflect real-world differences that may be seen in chest X-rays.

Training

In training your model, there are many parameters that can be tweaked to improve performance including:

Image augmentation parameters

Training batch size

Training learning rate

Inclusion and parameters of specific layers in your model

You will be asked to provide descriptions of the methods by which given parameters were chosen in the final FDA documentation.

Performance Assessment

As you train your model, you will monitor its performance over subsequence training epochs. Choose the appropriate metrics upon which to monitor performance. Note that 'accuracy' may not be the most appropriate statistic in this case, depending on the balance or imbalance of your validation dataset, and also depending on the clinical context that you want to use this model in (i.e. can you sacrafice high false positive rate for a low false negative rate?)

Note that detecting pneumonia is hard even for trained expert radiologists, so you should not expect to acheive sky-high performance. This paper describes some human-reader-level F1 scores for detecting pneumonia, and can be used as a reference point for how well your model could perform.

3. Clinical Workflow Integration

The imaging data provided to you for training your model was transformed from DICOM format into .png to help aid in the image pre-processing and model training steps of this project. In the real world, however, the pixel-level imaging data are contained inside of standard DICOM files.

For this project, create a DICOM wrapper that takes in a standard DICOM file and outputs data in the format accepted by your model. Be sure to include several checks in your wrapper for the following:

Proper image acquisition type (i.e. X-ray)

Proper image acquisition orientation (i.e. those present in your training data)

Proper body part in acquisition

4. FDA Submission

For this project, you will complete the following steps that are derived from the FDA's official guidance on both the algorithm description and the algorithm performance assessment. Much of this portion of the project relies on what you did during your EDA, model building, and model training. Use figures and statistics from those earlier parts in completing the following documentation.

1. General Information:

First, provide an Intended Use statement for your model

Then, provide some indications for use that should include:

Target population

When your device could be utilized within a clinical workflow

Device limitations, including diseases/conditions/abnormalities for which the device has been found ineffective and should not be used

Explain how a false positive or false negative might impact a patient

2. Algorithm Design and Function

In this section, describe your fully trained algorithm and the DICOM header checks that you have built around it. Include a flowchart that describes the following:

Any pre-algorithm checks you perform on your DICOM

Any preprocessing steps performed by your algorithm on the original images (e.g. normalization)

Note that this section should not include augmentation

The architecture of the classifier

For each stage of your algorithm, briefly describe the design and function.

3. Algorithm Training

Describe the following parameters of your algorithm and how they were chosen:

Types of augmentation used during training

Batch size

Optimizer learning rate

Layers of pre-existing architecture that were frozen

Layers of pre-existing architecture that were fine-tuned

Layers added to pre-existing architecture

Also describe the behavior of the following throughout training (use visuals to show):

Training loss

Validation loss

Describe the algorithm's final performance after training was complete by showing a precision-recall curve on your validation set.

Finally, report the threshold for classification that you chose and the corresponded F1 score, recall, and precision. Give one or two sentences of explanation for why you chose this threshold value.

4. Databases

For the database of patient data used, provide specific information about the training and validation datasets that you curated separately, including:

Size of the dataset

The number of positive cases and the its radio to the number of negative cases

The patient demographic data (as it is available)

The radiologic techniques used and views taken

The co-occurrence frequencies of pneumonia with other diseases and findings

5. Ground Truth

The methodology used to establish the ground truth can impact reported performance. Describe how the NIH created the ground truth for the data that was provided to you for this project. Describe the benefits and limitations of this type of ground truth.

6. FDA Validation Plan

You will simply describe how a FDA Validation Plan would be conducted for your algorithm, rather than actually performing the assessment. Describe the following:

The patient population that you would request imaging data from from your clinical partner. Make sure to include:

Age ranges

Sex

Type of imaging modality

Body part imaged

Prevalence of disease of interest

Any other diseases that should be included or excluded as comorbidities in the population

Provide a short explanation of how you would obtain an optimal ground truth

Provide a performance standard that you choose based on this paper: Link: https://arxiv.org/pdf/1711.05225

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Project Overview

In this project, you will apply the skills that you have acquired in this 2D medical imaging course to analyze data from the NIH Chest X-ray Dataset and train a CNN to classify a given chest x-ray for the presence or absence of pneumonia. This project will culminate in a model that can predict the presence of pneumonia with human radiologist-level accuracy that can be prepared for submission to the FDA for 510(k) clearance as software as a medical device. As part of the submission preparation, you will formally describe your model, the data that it was trained on, and a validation plan that meets FDA criteria.

You will be provided with the medical images with clinical labels for each image that were extracted from their accompanying radiology reports.

The project will include access to a GPU for fast training of deep learning architecture, as well as access to 112,000 chest x-rays with disease labels acquired from 30,000 patients.

Project Highlight

This project is designed to give you hands-on experience with 2D medical imaging data analysis and preparation of a medical imaging model for regulatory approval.

Upon completion of this project, you would be able to:

- recommend appropriate imaging modalities for common clinical applications of 2D medical imaging

- perform exploratory data analysis (EDA) on medical imaging data to inform model training and explain model performance

- establish the appropriate ‘ground truth’ methodologies for training algorithms to label medical images

- extract images from a DICOM dataset

- train common CNN architectures to classify 2D medical images

- translate outputs of medical imaging models for use by a clinician

- plan necessary validations to prepare a medical imaging model for regulatory approval

Project Steps

This project has the following steps.

1. Exploratory Data Analysis

2. Building and Training Your Model

3. Clinical Workflow Integration

4. FDA Preparation

For this project, you will work in the Jupyter GPU workspace provided for you. You can also find the notebooks containing the necessary starter code in the workspace.

You may also download all of the files for the project directly from this repo(opens in a new tab). This workspace contains:

EDA.ipynb: This is the file you will be performing the EDA.

Build and train model.ipynb: This is the file you will be building and training your model.

Inference.ipynb: This is the file you will be performing clinical workflow integration.

.dcm files: They are the test files to test the clinical workflow integration.

sample\_labels.csv: This is the file that should be used to assess images in the pixel-level.

FDA\_Submission\_Template.md: This is the template for you to create the FDA submission. Please copy the template into your choice of editor. Finish the documentation, save it as a .pdf file, and upload.

Note: The NIH data for EDA and training is mounted in the Udacity Jupyter GPU workspace provided to you along with the code to load the data. Alternatively, you can download the data from the kaggle website(opens in a new tab) and run it locally. You are STRONGLY recommended to complete the project using the Udacity workspace since the data is huge, and you will need GPU to accelerate the training process.

Detailed instruction for this project is provided in this README(opens in a new tab) file.

Evaluation

Your project will be reviewed by a Udacity reviewer against the project rubric(opens in a new tab). Be sure to review this rubric thoroughly and self-evaluate your project before submission. All criteria found in the rubric must be meeting specifications for you to pass.

Submission Files

Following files would be needed for evaluation:

EDA.ipynb notebook file with all questions answered and all code cells executed and displaying output.

Build and train model.ipynb notebook file with all questions answered and all code cells executed and displaying output.

A .h5 or .json file contains your final model architecture.

A .h5 or .json file contains your final model weights.

Inference.ipynb notebook file with all questions answered and all code cells executed and displaying output.

FDA\_submission.pdf file with a report that describes the algorithm and performance.

Please upload the FDA\_submission.pdf to the workspace and submit your project through the workspace.

**Course 3: AI to 3D Medical Imaging**

Part 1: Introduction

Course objectives:

- Understand what 3D medical images are, who uses them, and for what purposes

- Perform exploratory data analysis on 3D image datasets in common formats such as DICOM and NIFTI

- Apply popular machine learning algorithms for both classification and segmentation tasks using real-world medical imaging datasets

- Learn to integrate trained models into a clinical imaging environment and troubleshoot your deployments

- Provide input into the algorithm validation process as required for field deployments

AI: term definition:

“An attempt will be made to find how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves. … For the present purpose the artificial intelligence problem is taken to be that of making a machine behave in ways that would be called intelligent if a human were so behaving.”

McCarthy, Shannon, Minsky et al. A proposal for the Dartmouth summer research project on artificial intelligence, 1955

AI engineers are like full-stack engineers that need to think about all aspects of workflow from the data, front-end usage and back-end implementation. Machine learning is considered a subpart of AI that is data-focused on improving algorithms that do not need to be hard-coded, but can evolve with the data that is provided.

FDA algo approvals since 2016. 100+ radiology startups for AI algorithms.

Stakeholders for AI for medical imaging: hospitals/medical centers, startups, large medical software vendors

Tools for the course:

- Machine learning tools: Python 3.7/Jupyter, PyTorch 1.3, Numpy 1.18

- Imaging tools: Viewers: Microdicom, Radiant, 3D slicer

- Clinical network simulation/debugging: Orthanc, OHIF, DCMTK

Packages for local implementation: PyTorch (with CUDA), nibble, matplotlib, bumpy, Pillow, tensor board, 3D Slicer, DMCTK tools

Part 2: Medical Imaging

- 3D imaging tasks: multi-planar reconstruction, 3D reconstruction, windowing, registration

Lesson Summary

In this lesson, we will have Mazen share his perspective on the following:

• What are 3D medical images?

• Who uses 3D medical images?

• Why are they being used?

• Some example clinical scenarios

After that, we will have you perform an exercise on picking a suitable problem for an Al project by tapping into publicly available medical resources.

Then, we will go into some technical details and cover:

• Physical principles of CT scanners and then perform an exercise on computing a sinogram

• Physical principles of MR scanners

• Cover basic 3D imaging tasks:

• Multi-planar reconstruction

• 3D reconstruction

• Windowing

• Registration

We will finish the lesson by having you perform an exercise where you will write code to do a 3D reconstruction of a 3D medical volume.

Why 3D medical images:

- better at answering “IS” there a finding, better localization in 3D spaces, what tissue types

- parameters: contrast resolution, spatial resolution, invasiveness, radiation dose, cost

Contrast Resolution: differences in image intensity

- each imaging modality has its own intrinsic contrast resolution, which can be adjusted by tweaking certain imaging parameters prior to image acquisition. (Only intrinsic parameters)

- contrast agent/media can also be introduced to improve resolution. CT: iodine-based agents, MRI: gadolinium-based agents.

- following IV administration, these agents follow circulatory system, first returning to the heart, pumped through the arteries, delivered to the organs, returned to the veins, and eventually excreted through the urinary system and hepatobiliary system.

- Above cycle can be used to capture images at different time points to highlight different anatomical structures. Ex: CT of abdominal can take pictures prior to contrast injection, 20 seconds later when the artery stand out most, 75 seconds later when the liver and venous system are highlighted, finally a delayed time point which can be useful for diagnosing some diseases.

- ask whether images are non-contrast, or post-contrast, and if a contrast agent was administered which phase the patient was imaged in.

Spatial Resolution: differences between small objects

- CT-Face vs. CT-Temporal Bone (have different spatial resolution for middle ear)

- spatial resolution refers to what the parameters and FOV will be and how it will affect ability to resolve different anatomical structures

Application:

- diagnosis made clinically or based on history, can be further informed using imaging studies to refine the diagnosis, more precisely estimate the severity of disease, and attempt to predict future progression.

- Example: Alzheimer’s Disease: volumetric MRI with quantification of key brain regions can be used to more precisely estimate the severity of the disease, or identify any additional contributing factors to the patient’s memory loss.

- Final project will involve the hippocampus and segmenting this structure in the context of AD

- Medical imaging can be used to guide therapy, ex: radiation therapy where CT/MRI can be used to contour regions of interests, planning the distribution and quantity of radiation with higher precision.

Bayesian Theorem: describes the probability of an event based on prior knowledge of conditions that might be related to the event. Expressed in terms of odds in medicine.

Prior Odds x Likelihood Ratio = Posterior Odds

Prior Odds: refers to odds prior to diagnostic test in question (i.e. local population prevalence)

Likelihood Ratio: test diagnosis performance metric that needs to be calculated and ranges from 0 to infinity where 1 means prior odds is unchanged

Posterior Odds: belief about a disease after factoring in our diagnostic test.

High likelihood ratio would confirm disease by significantly increasing odds, while low likelihood ratio test would exclude disease by significantly decreasing odds.

Exercise 1: Choosing a clinical problem and framing it as a machine learning task.

In this exercise, you will practice how to choose a clinical problem to tackle, explore how to research some basic facts about the disease, and frame the clinical problem as a machine learning task.

Part 1: Choosing a clinical case.

Exciting clinical problems abound in the realm of Healthcare AI. To help you narrow down some potential ideas, we will use the American College of Radiology Data Science Institute (ACR DSI) curated list of use cases. Alternatively, you may explore any clinical problem of your choosing without using the database if you already had something in mind. Please note that some use cases in the ACR DSI database can seem overwhelming or obscure without some background medical knowledge. If this is the case, simply choose something that’s more intuitive. A few good starting examples include:

Acute Appendicitis.

Aging Brain – Dementia: https://www.acr.org/Data-Science-and-Informatics/AI-in-Your-Practice/AI-Use-Cases/Use-Cases/Aging-Brain---Dementia

Incidental Pulmonary Nodules on CT. Your task for this first part is to choose a case. If you choose to use the ACR DSI database, submit the link that points to your use case. If using any other resources, please submit an image capture or PDF of a medical or AI journal article that explores this problem within the context of machine learning.

Part 2: Background research.

To deliver an algorithm that performs well in a clinical setting, it can be extremely beneficial to have some rudimentary understanding of the disease or situation you are addressing. To do so, engaging in some background research to build your foundation will be crucial. For this part of the exercise, find a review article from the PubMed database describing the problem. Ask yourself, “what is the current gold standard for confirming or ruling out the presence of this disease/state”. Understanding the current gold standard/ground truth test will be a critical benchmark when testing any algorithm you develop. Submit a screen capture or PDF of the article to receive credit for this portion.

Part 3: Framing the problem as a machine learning task.

With some background knowledge of the disease or condition at hand, as well as how it is currently ruled in or out, consider how the task could be framed as a machine learning problem. We will be going over this in much more detail in the respective lesson of this course, but some early exposure will be useful. For example, solving the problem of autonomously detecting one or more lung nodules could be framed as an object detection task. Measuring changes in tumor volume over time may be best framed as a segmentation task. Etc. Submit a short paragraph detailing how you would frame your clinical problem of choice as a machine learning task, and explain your rationale.

Below you will find a workspace with a Jupyter Notebook where you can record the answers to each part of this exercise listed above.

Example from course instructor:

Part 1: Choosing a clinical case.

COVID-19 Compatible Chest CT Pattern(opens in a new tab).

Part 2: Background research.

Focused on Performance of radiologists in differentiating COVID-19 from viral pneumonia on chest CT(opens in a new tab) paper by Bai, H.X., Hsieh, B., et. al. for my background research.

Part 3: Framing the problem as a machine learning task.

Per the ACR-DSI, the assigned task is to provide a likelihood of a diagnosis compatible with COVID-19 using chest CT data. I believe an initial step in solving this clinical problem, determining whether any abnormal findings are present on a chest CT, would be well-framed as an object detection task. The output would be a bounding box delineating the areas of abnormality that could indicate viral pneumonia. Based on recent literature, such findings could include consolidation, bilateral and peripheral disease, linear opacities, “crazy-paving” patterns, and the “reverse halo” sign. While these findings are not-specific for COVID-19, their detection could guide the ordering clinician to be more vigilant in follow-up testing for the disease. A negative STAT rapid influenza/RSV PCR tests and positive Real-Time Reverse Transcriptase Polymerase Chain Reaction (rRT-PCR) would confirm the diagnosis.

Hounsfield Units/Scale or CT numbers:

- scanner measures attenuation of X-rays which largely depend on the physical density of material that X-rays are passing through.

- backprojection process gives values that represent attenuation at a given point in space

- these values are calibrated to a range of ~1000 (least dense) to ~3000 (densest) and are typically represented by grayscale values on screen.

- bone: 400-1000, soft tissue: 400 to 80, water: 0, fat: -60 to -100, lung: -400 to 600, air: -1000

MR Scanners:

- Pulse sequence: the combination of a static field, gradient field, and RF pulse that can be varied with time defines a pulse sequence. Once the induced currents are measured as a result of proton processing and relaxation, the measurements can be represented as values in a physical construct known as k-space.

- Once we have k-space data, it undergoes a series of transformations in a reconstructed pipeline, to obtain the final image which represents exact measured signal intensities in the physical space.

- MRI: T1, T2-weighted sequences, FLAIR, DWI, DTI, PD are all different pulse sequences

- some sequences are with contrast media

- unlike CT data, MR values do not correlate with tissue density well; they do not have same consistent meaning

- raw data potentially offers lot of opportunities for data scientists.

Windowing: linear transformation of a section of original image color space, to convert to grayscale.

Multi-planar Reconstruction (MPR): one plane is imaged; the others are reconstructed; usually the non-imaging planes will have lower resolution or will have artifacts. Refers to extraction of non-primary imaging planes from a 3D volume.

3D reconstruction: surface mesh vs. volume mesh; constructing a 3D model from multiple slices of 3D medical imaging data.

Registration: bringing two different images into the same coordinate space, lining images up together in 3D space; ex: different modalities or different time points. Shift voxels from one image to another image. Rigid registration refers to transformation where only rotation and translation is applied to the entire volume. Affine registration allows scaling and addition to transformation and rotation. Straight lines of moving image remain straight but sizes are not preserved. Deformability registration refers to the process of applying transform to each single point of the moving volume individually. Example deformable registration can be used to map the MRI image on top of the CT image.

Part 3: 3D imaging exploratory data analysis

DICOM: Digital Imaging and Communications in Medicine (free and open standard)

- used by all scanners and radiological modalities

- website: https://www.dicomstandard.org/

- Top level: Patient -> Study (One imaging session with scanner) -> Series (one acquisition sweep within study) -> Instance (one image within series: pixel data & metadata or data elements)

- DICOM: defines both the standard for how medical imaging data is transmitted through networks and a way to store medical images

- DICOM - “Digital Imaging and Communications in Medicine”. A standard that defines how medical imaging (primarily) data is stored and moved over the network.

- Structure: DICOM File = Meta Information + DICOM data set (text-based data elements + pixel data elements: frame 1, 2, etc)

- a lot of the same data is saved in meta information and dicom data elements

- SOP Classes(Service Object Pair): Each have different identifiers to see what image one is looking at

- DICOM Data Elements: meta data about the file, key value pairs that hold information about the files. Includes information on Image Type, Creation time, Content Date, Acquisition Date/Time, Study Time, etc. These are not mandatory and may or may not be in the DICOM file.

-VR (value representation) column: different abbreviations for type of data (ex: string, etc)

- Data Elements: have Study UID, Series UID, Instance UID (unique identifier). Patient ID is not a mandatory element.

- DICOM standard rules can be changed for specific reasons, thus need to study each separately.

- DICOM standard: will also show the modules or groups of DICOM data elements that are needed or essential for an MR image. Mandatory: M label.

- NIFTI: used for imaging competitions or preparing DICOM datasets for machine learning pipelines.

Terminology:

SOP - Service-Object Pair. DICOM standard defines the concept of an Information Object, which is the representation of a real-world persistent object, such as an MRI image (DICOM Information Objects consist of Information Entities). The standard also defines the concept of Services that could be performed on Information Objects. One such service is the Storage service (we will touch on others later in the course), and a DICOM image stored as a file on a file system is an instance of Storage service performed on an Image Information Object. Such Service-Object Pairs have unique identifiers that help unambiguously define what type of data we are dealing with. A list of SOP Classes can be found in Part 4 of the Standard(opens in a new tab). This list is a useful reference for all possible data types that could be stored per the DICOM standard.

Data Element - a DICOM metadata “field”, which is uniquely identified by a tuple of integer numbers called group id and element id. The convention is to write the element identifier as group id followed by the element id in parentheses like so: (0008,0020) - this one is the DICOM Element for Study Date. DICOM data elements are usually called “tags”. You can find the list of all possible DICOM tags in Part 6, Chapter 6 of the standard(opens in a new tab).

VR - Value Representation. This is the data type of a DICOM data element. DICOM standard imposes some restrictions on what form the data can take. There are short strings, long strings, integers, floats, datetime types, and more. You can find the reference for DICOM data types in Part 5, Section 6 of the standard(opens in a new tab)

Data Element Type - identifiers that are used by Information Object Definitions to specify if Data Elements are mandatory, conditional or optional. Data Element Type reference can be found in Part 5, Section 7 of the standard(opens in a new tab)

DICOM Information Object - representation of a real-world object (such as an MRI scan) per DICOM standard.

IOD - Information Object Definition. Information Object Definition specifies what metadata fields have to be in place for a DICOM Information Object to be valid. Scanner manufacturers follow the relevant parts of the DICOM standard when saving the digital data acquired by the scanner. When parsing DICOM data, it is often useful to reference the relevant IODs to see what data elements could be expected in the particular class of information objects, and what they mean. For example, in Part 3 of the standard, you can find MR Image IOD(opens in a new tab) and CT Image IOD(opens in a new tab) which we will use in this course quite a bit. You might have noticed that the table with all DICOM data elements does not really provide any description of what these elements mean. The reason for that is that elements may mean slightly different things depending on what Information Object Definition uses them, therefore, to find the real meaning of the element you need to look them up in the respective IOD.

NIFTI File Format

NIFTI - Neuroimaging Informatics Technology Initiative, is an open standard that is used to store various biomedical data, including 3D images.

- started for MRI data, then evolved to store any type of biomedical imaging data.

- stores entire series in a single file; so do not have multiple files representing a 3D volume

- NIFTI is not generated by scanners, unlike DICOM file format

- NIFTI will not have all the scanner metadata that DICOM does

- widely used in imaging competitions and Machine Learning

- does have metadata

- also has a website for the standard: https://nifti.nimh.nih.gov/nifti-2/

- orientation information is stored differently from DICOM (different coordinate system)

- stores pixels/voxels in a single file filed

- NIFTI stores information in variable units of measurement (DICOM only millimeters)

• Background and history of the NIFTI: https://nifti.nimh.nih.gov/background/

• The most "official" reference of NIFTI data fields could be found in this C header file, published on the standard page: https://nifti.nimh.nih.gov/pub/dist/src/niftilib/nifti1.h or on this, slightly better-organized page: https://nifti.nimh.nih.gov/nifti1/documentation/nifti1fields

• A great blog post on NIFTI file format: https://brainder.org/2012/09/23/the-nifti-file-format/

Viewer:

- MicroDicom (Windows OS): free but for windows

- 3D Slicer (Windows, Linux and Mac): open-source modular 3D medical image viewer that can be used to view both DICOM and NIFTI.

DICOM Parameters: Coordinate System:

- patient coordinate system

- scanner coordinate system

- DICOM has orientation of image relative to patient: Image Position Patient (IOP): position of top-left corner, in patient coordinates, Image Orientation Patient (IPP): orientation of image

- load image into 3D Slicer: orientation should be what is expected when processing dataset

- Pixel Spacing: x and y dimension size (can be different)

- Slice Thickness: z dimension (may not be available in DICOM)

NIFTI Parameters:

- pixdim field, xyzt\_units is for units of measurements

Orientation parameters

For DICOM two parameters that define the relative position of a 2D in the 3D space would be:

(0020,0037) Image Orientation Patient - a parameter that stores two vectors (directional cosines to be precise) that define the orientation of the first row and first column of the image.

(0020,0032) Image Position Patient - a parameter that stores x, y, and z coordinates of the upper left-hand corner of the image.

Both of these are Type 1 (mandatory) parameters for MR and CT IODs, so it is generally safe to rely on them.

For NIFTI, the same purpose is served by srow\_\*, qoffset\_\* vectors.

Physical spacing parameters

(0028,0030) Pixel Spacing - two values that store the physical distance between centers of pixels across x and y axes.

(0018,0050) Slice Thickness - thickness of a single slice. Note that this one is a Type 2 (required, but can be zero) parameter for CT and MR data. If you find those unavailable, you can deduce slice thickness from IPP parameters. This can happen if your volume has non-uniform slice thickness.

Photometric parameters

There are quite a few of those, as DICOM can store both grayscale and color data, so lots of parameters deal with color palettes. CT and MR images usually have monochrome pixel representation (defined by tag (0028,0004) Photometric Interpretation).

Most notable ones of this group are:

(0028,0100) Bits Allocated - parameter that defines the number of bits allocated per pixel (since we have CPUs that operate in bytes, this parameter is always a multiple of 8).

(0028,0101) Bits Stored - parameter that defines the number of bits that are actually used - quite often, you could see Bits Allocated set to 16, but Bits Stored set to 12.

Image size parameters

Of worthy mention are parameters that define the size of the 3D volume. There are Type 1 parameters that define the width and height of each 2D slice:

(0020,0010) Rows - this is the height of the slice, in voxels

(0020,0011) Columns - width of the slice, in voxels

Both of these need to be consistent across all DICOM files that comprise a series.

Note that there isn't really anything in DICOM metadata that has to tell you how many slices you have in the series. There are tags that can hint at this (like (0054,0081) Number of Slices, or (0020,0013) Instance Number), but none of them are mandatory, Type 1 tags for CT or MR data. The most reliable way to determine the number of slices in the DICOM series is to look at the number of files that you have, and ideally validate that they make up a correct volume by checking for the consistency of IPP values.

Further Resources

If you want to dive deeper into the subjects of coordinate spaces for medical images, and parameters of DICOM files in general, some useful resources:

• Section on IPP and IOP parameters in the DICOM standard: http://dicom.nema.org/medical/dicom/2020a/output/chtml/part 03/sect C.7.6.2.html

• A solid explanation of how coordinate systems work in NIFTI: https://nipy.org/nibabel/coordinate systems.html

• A company called Innolitics (a vendor of various DICOM software) maintains a great reference of the DICOM standard which sometimes could be quite a bit more convenient than the official standard: https://dicom.innolitics.com/ciods

Voxel spacing

DICOM voxels do not have to be perfect cubes (as they are in many computer vision problems). There are DICOM Data Elements that will tell you what exactly are the dimensions of voxels. The most important ones are Pixel Spacing and Slice Thickness. However, there are others, and if your project involves measuring things, make sure you get the transformation right by closely inspecting the tags in your dataset and comparing them with the list of elements in the IOD table for the respective modality.

Data ranges

We have seen how with CT, you may have data in your dataset that will represent synthetic material or items artificially added by scanners. It is always a good idea to see if there is something outstanding in the image you are dealing with and if it represents something that you need to think about in your downstream processing.

Conversions between DICOM values and screen space are particularly important if you are planning to visualize slices for any kind of diagnostic use or overlay them on top of diagnostic information. We have not really touched the aspects of visualization other than being mindful of bit depth and doing our own windowing, but DICOM images contain quite a lot of information that defines how exactly you are expected to map the data to the screen colorspace. If you are interested in exploring this further or need to accurately represent the data, take a closer look at elements in DICOM's ImagePixel module. Things like Pixel Representation, Photometric Interpretation, Rescale Slope, Rescale Intercept and many others define how values should be transformed for accurate representation.

Methods for dataset analysis basically boil down to using the same tricks as you’d do for individual volume analysis and being on the lookout for inconsistencies in data.

Inconsistencies usually boil down to two classes:

Clinical anomalies - the things related to either anatomical anomalies like missing organs, pathologies like tumors or implants such as limb prosthesis, ports/cannulas, surgical implants, presence of contrast media, etc. Sometimes these things can result in artifacts in the images, so it’s good to be aware of them

Informatics anomalies - things related to specifics of data acquisition or variations in DICOM encoding coming from different scanners. These would be things like slice spacing consistency, image dimensions, variations in photometric encoding, etc

Basic knowledge of DICOM and intuition for what things could go wrong are always useful when analyzing the datasets. I will post some examples of great dataset EDA at the end of this lesson as well.

**Part 4: 3D Medical Imaging: Deep Learning Applications:**

• Use cases for 3D medical image classification and object detection

• Distinguish between 2D and 2.5D methods for 3D volume classification

• Identify the use cases for segmentation

• Apply the U-Net algorithm to train a machine learning model for segmentation

• Understand Ground Truth for Segmentation Understand common segmentation metrics: Dice, HD

Classification Use Cases: where AI can be applied

- Acute/Emergent Diagnosis (algo at point of image acquisition for list prioritization, or resource-limited areas)

- Screening: automated detection for screening as these are labor intensive

- Incidental Findings: ex: incidentalomas (ex: cysts)

Point of View: “when choosing a medical imaging problem to be solved by machine learning, it is tempting to assume that automated detection of certain conditions would be the most valuable thing to solve. However, this is not usually the case. Quite often detecting if a condition is present is not so difficult for a human observer who is already looking for such a condition. Things that bring most value usually lie in the area of productivity increase. Helping prioritize the more important exams, helping focus the attention of a human reader on small things or speed up tedious tasks usually is much more valuable. Therefore it is important to understand the clinical use case that the algorithm will be used well and think of end-user value first.”

2D vs. 2.5D vs. 3D Convolution:

2D Convolution is an operation visualized in the image above, where a convolutional filter is applied to a single 2D image. Applying a 2D convolution approach to a 3D medical image would mean applying it to every single slice of the image. A neural network can be constructed to either process slices one at a time, or to stack such convolutions into a stack of 2D feature maps. Such an approach is fastest of all and uses least memory, but fails to use any information about the topology of the image in the 3rd dimension.

2.5D Convolution is an approach where 2D convolutions are applied independently to areas around each voxel (either in neighboring planes or in orthogonal planes) and their results are summed up to form a 2D feature map. Such an approach leverages some 3-dimensional information.

3D Convolution is an approach where the convolutional kernel is 3 dimensional and thus combines information from all 3 dimensions into the feature map. This approach leverages the 3-dimensional nature of the image, but uses the most memory and compute resources.

Understanding these is essential to being able to put together efficient deep neural networks where convolutions together with downsampling are used to extract higher-order semantic features from the image.

Practicing Convolutions: Exercise

PyTorch: python library for creating CNN

Segmentation: Pixels/Voxels belong to which object

Use Case: Oncology

* Automated segmentation for tracking volume of mass x time
* Volumetric measurements

Use Case: Alzheimer’s disease

* Segmentation of Hippocampi relative to age matched cohorts can provide information on disease severity

Use Case: Radiation Therapy Planning

* Contours are drawn around tumor volume avoiding critical structures
* Automated segmentation can accelerate this process
* Here, automated segmentation to delineate the volume of tissue that’s required for analysis during the pipeline can save time and effort compared to manually generating this data in a large dataset

Segmentation Architectures

* Key concept: for a segmentation CNN, often add an upsampling path to the downsampling path that is commonly employed in CNN for classification
* This is because in the end, want to classify every single voxel of an image that is treated as a classification problem (i.e. dance prediction)
* Example of a segmentation network is U-NET (published in 2015), successful for pathological slides and radiological imaging
* U-NET network: series of convolutional layers which reduce the dimensions of an image and stack together several convolutional maps and then a max pooling layer until reach the bottleneck of the U, where the channels increase. Then start an up convolution step which concatenates the results of the corresponding step from the down-sampling path which adds robustness to the network and gives it interesting context on both high resolution but less contextual features and lower resolution, but features bearing higher context.
* Exercise: Train a U-NET algorithm on a smaller dataset to see how it converges on predicting the shape of the a spleen on a CT scan of an abdomen
* A U-Net architecture has been very successful in analyzing 3D medical images and has spawned multiple offshoots. You will get a chance to get more familiar with it in the exercise that follows, but if you would like to understand the principles better, I recommend that you check out the webpage on U-net created by one of the authors of the original paper, Olaf Ronneberger: https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/index.html(opens in a new tab). You will find the link to the original paper and a few materials explaining how and why this architecture works.

Ground Truth for Segmentation

* There are very few clinical workflows that create ground truth for medical imaging (ex: quantitative radiology).
* Need to consider using clinicians to establish ground truth but this depends on level of agreements especially for edges where it can be difficult or hard to reach agreement among labelers
* Need to have a common contouring protocol as there may be variability in responses
* Segmentation Tooling for creating ground truth: example is 3D Slicer, but all the tooling requires anatomy knowledge and clinical context

Evaluating Model Performance of Segmentation Algorithms

* Cross-Entropy Loss: tends to favor the dominating class (bias)
* Dice Similarity Coefficient (DSC): fairly popular choice for measuring performance of segmentation algorithms
* Jaccard index: also fairly popular and similar to DSC. Generally need to look at either DSC or Jaccard Index. However, both DSC and JI have shortcomings in that they do not tell you exactly how structures are different. Irregular shapes like vascular structures is not great for DSC
* Hausdorff Distance: Can be used instead of DSC or JI. Penalizes when the model tends to create long off-shoots or the ground truth structures are highly irregular. However, reqiuires intensive calculations and then need to take a mean or average of distances.
* Sensitivity & Specificity: compare voxels that are correct vs. incorrect; usually need to be presented together

Clinical evaluation of Segmentation Algorithms:

* Likelihood Ratio: diagnostic performance metric used to determine whether a test usefully changes the probability that a condition or disease is present. This can be calculated based on sensitivity / specificity.
* The likelihood ratio for a diagnostic test result can be calculated if the predictive characteristics (sensitivity and specificity) of that test are known. Likelihood ratios are known for common diagnostic tests performed by humans (e.g., correctly identifying viral pneumonia from chest CT scans).
* This means that for example, your ML segmentation algorithm may be measuring the volume of a specific anomaly in the lung very accurately, but this measurement, while important to quantify the degree of lung involvement by some disease state, may be not specific at all for predicting whether that state is due to a viral pneumonia (e.g., presence of such anomalies could mean viral pneumonia, bacterial pneumonia or non-infectious causes like hemorrhage or edema). Thus, your algorithm with high Dice scores may end up being not very useful to solve a clinical task if the goal is a specific diagnosis.

Other Use Cases for Deep Learning

* Registration as optimization problem
* Automatic discovery of optimal reconstruction pipelines
* Image retrieval: finding similar images which have clinical context
* Ground Truth augmentation: for imbalanced datasets such as rare diseases, generating simulation positive cases to create a balanced dataset of a rare disease

• Using deep learning to increase the resolution of low-res scans: Chaudhari AS, Fang Z, Kogan F, et al. Super-resolution musculoskeletal MRI using deep learning. Magn Reson Med.

2018:80(5):2139-2154. doi:10.1002/mrm.27178

• GANs for synthetic MRI: Frid-Adar, M., Diamant, L., Klang, E., Amitai, M., Goldberger, J:, & Greenspan, H. (2018). GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification. Neurocomputing, 321, 321-331. doi: 10.1016/j.neucom.2018.09.013

• A survey of deep learning methods for medical image registration: Haskins, G., Kruger, U. & Yan, P. Deep learning in medical image registration: a survey. Machine Vision and Applications 31, 8 (2020). https://doi.org/10.1007/s00138-020-01060-x

• Overview of opportunities for deep learning on MRIs: Lundervold, A. S., & Lundervold, A. (2019). An overview of deep learning in medical imaging focusing on MRI. Zeitschrift Für Medizinische Physik, 29(2), 102-127. doi: 10.1016/j.zemedi.2018.11.002

Books

Some resources readily available online for free will help you grasp the basic concepts of computer vision and overall machine learning.

• https://d2l.ai/ - deep learning with a special section on computer vision by Alexander Smola et al. Alexander has a strong history of publications on machine learning algorithms and statistical analysis and is presently serving as a director for machine learning at Amazon Web Services in Palo Alto, CA

• http://www.mbmlbook.com/ - a book on general concepts of machine learning by Christopher Bishop et al. Christopher has a distinguished career as a machine learning scientist and presently is in charge of Microsoft Research lab in Cambridge, UK, where I had the honor to work on project InnerEye for several years.

More notable papers

• If you're curious about segmentation space specifically, you may appreciate a foray into non-ML-based methods for segmentation. A couple of papers that can provide an introduction into that space are: • Boykov, Y., & Jolly, M.-P. (2000). Interactive Organ Segmentation Using Graph Cuts. Medical Image Computing and Computer-Assisted Intervention - MICCAl 2000 Lecture Notes in Computer Science, 276-286. doi: 10.1007/978-3-540-

40899-4 28

• Probabilistic Graphical Models for Medical Image Segmentation

• This GitHub repo provides an excellent overview of CNN-based seg methods for general image domain: https://github.com/mrgloom/awesome-semantic-segmentation

**Part 5: Deploying AI Algorithms in Real World Scenarios**

In our final lesson, we will focus on what it takes to build an Al algorithm like some of those we talked about in the previous lesson into a realworld system. We will discuss the following:

• Basics of DICOM networking

• How hospital networks operate and where would Al algorithms fit In

• Requirements for integration of Al algorithms

• Tools for simulating and debugging clinical environments:

• scripting via DCMTK

• OHIF - the zero-footprint medical image viewer

• A deeper-dive into 3DSlicer for annotation

• Medical imaging viewer as radiologist's tool - how radiologists interact with viewers

• Medical Device Regulations with the US FDA as an example. We will talk about the regulation of medical devices by the government and how it applies to Al algorithms.

• Data privacy, HIPAA, and anonymization.

The content of this lesson is meant to give you an idea of what it takes to bring an Al system into the real world. This is where the job of data scientist typically ends and collaboration of cross-disciplinary teams is involved. However, your activities as an Al engineer should be informed by what happens with the system in the real world, and likewise, there are aspects of operation and deployment where the input of an Al engineer is instrumental. Here we try to help you understand what kinds of things to pay attention to.

When it comes to moving medical images around the hospital, the DICOM standard comes to the rescue. Alongside the definition of the format for storing images and metadata (which we have looked at in detail in previous lessons, it defines the networking protocol for moving the images around.

DICOM Networking - DIMSE

DIMSE: DICOM Message Service Element

Application Entities use DIMSE to communicate over LAN

To integrate AI systems into clinical workflows need to know the networking and commands needed to transfer or find DICOM files between different entities

To summarize, here are the key things and terms to remember about DICOM networking:

1. There are two types of DICOM networking: DIMSE (DICOM Message Service Element) and DICOMWeb. The former is designed to support data exchange in protected clinical networks that are largely isolated from the Internet. The latter is a set of RESTful APls (link to the Standard) that are designed to communicate over the Internet.

DIMSE networking does not have a notion of authentication and is prevalent inside hospitals.

2. DIMSE networking defines how DICOM Application Entities talk to each other on protected networks.

3. DICOM Application Entities that talk to each other take on roles of • Service Class Providers which are an an AE that provides services over DIMSE network and

• Service Class Users which is an Al that requests service from an SCP

4. SCPs typical respond to requests and SCUs issue them

5. Full list of DIMSE services could be found in the Part 7 of the DICOM Standard, ones that you are most likely run into are:

• C-Echo - "DICOM ping" - checks if the other party can speak DICOM

• C-Store - request to store an instance

6. An Application Entity (AE) is an actor on a network (e.g. a medical imaging modality or a PACS) that can talk DIMSE messages defined by three parameters:

• Port

• IP Address

• Application Entity Title (AET) - an alphanumeric string

Definitions:

PACS - Picture Archiving and Communication System. An archive for medical images. A PACS product typically also includes "diagnostic workstations" - software for radiologists that is used for viewing and reporting on medical images.

VNA - Vendor Neutral Archive. A PACS that is not tied to a particular equipment manufacturer. A newer generation of PACS. Often deployed in a cloud environment.

EHR - Electronic Health Record. A system that stores clinical and administrative information about the patients. If you've been to a doctor's office where they would pull your information on a computer screen and type up the information - it is an EHR system that they are interacting with. EHR system typically interfaces with all other data systems in the hospital and serves as a hub for all patient information. You may also see the acronym "EMR", which typically refers to the electronic medical records stored by the EHR systems.

RIS - Radiology Information System. Think of those as "mini-EHRs" for radiology departments. These systems hold patient data, but they are primarily used to schedule patient visits and manage certain administrative tasks like ordering and billing. RIS typically interacts with both PACS and EHR.

In addition to DICOM protocol there are two more (among many) that you might run into:

HL7 - Health Level 7. A protocol used to exchange patient data between systems as well as data about physician orders (lab tests, imaging exams) FHIR - Fast Healthcare Interoperability Resources. Another protocol for healthcare data exchange. HL7 dates back to the '80s and many design decisions of this protocol start showing their age. You can think of FHIR as the new generation of HL7 built for the open web.

DCMTK: open-source project used for parsing DIOM files and simulating medical imaging networks

Here, we used some of the tools from the DCMTK toolkit to emulate the operation of a DICOM network. We have done the following:

• Used demdump tool to view the DICOM metadata of a DICOM file • Used storecsp command to bring up an SCP listening to incoming C-STORE requests. The following is the command line that we used: storescp 109 -v -aet TESTSCP -od . --sort-on-studyuid st

. This starts listening on port 109, with verbose logging, with AE Title "TESTSCP", putting all incoming files into the current directory and organizing studies into directories named as study instance UIDs, with prefix st

• Used echoscu command to verify that our SCP instance is alive and listening by running the following on the command line: echoscu localhost 109 -v

• Used storescu command to issue a set of C-STORE requests to our SCP and send several DICOM studies. We used the following command to accomplish this: storescu localhost 109 -v -aec TESTSCU +r +sd . . Here, -aec parameter specifies the AE title that our SCU will use to identify itself (some SCPs might only receive data from known AE titles); +r parameter tells our tool to process directories recursively and +sd parameter specifies a directory to send.

3DSlicer: can load dicom or nifti files

OHIF tool: JavaScript, single web-page application.

OHIF website: http://ohif.org/(opens in a new tab)

Cornerstone GitHub repository: https://github.com/cornerstonejs/cornerstone

Search and Example 510K for FDA

<https://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfPMN/pmn.cfm>

https://www.accessdata.fda.gov/cdrh\_docs/pdf19/K192437.pdf

**Final Project:**

**Quantifying Hippocampus Volume for Alzheimer's Progression**

**Background**

Alzheimer's disease (AD) is a progressive neurodegenerative disorder that results in impaired neuronal (brain cell) function and eventually, cell death. AD is the most common cause of dementia. Clinically, it is characterized by memory loss, inability to learn new material, loss of language function, and other manifestations.

For patients exhibiting early symptoms, quantifying disease progression over time can help direct therapy and disease management.

A radiological study via MRI exam is currently one of the most advanced methods to quantify the disease. In particular, the measurement of hippocampal volume has proven useful to diagnose and track progression in several brain disorders, most notably in AD. Studies have shown a reduced volume of the hippocampus in patients with AD.

The hippocampus is a critical structure of the human brain (and the brain of other vertebrates) that plays important roles in the consolidation of information from short-term memory to long-term memory. In other words, the hippocampus is thought to be responsible for memory and learning.

According to Nobis et al., 2019, the volume of hippocampus varies in a population, depending on various parameters, within certain boundaries, and it is possible to identify a "normal" range taking into account age, sex and brain hemisphere.

There is one problem with measuring the volume of the hippocampus using MRI scans, though - namely, the process tends to be quite tedious since every slice of the 3D volume needs to be analyzed, and the shape of the structure needs to be traced. The fact that the hippocampus has a non-uniform shape only makes it more challenging.

As you might have guessed by now, we are going to build a piece of Al software that could help clinicians perform this task faster and more consistently.

You have seen throughout the course that a large part of Al development effort is taken up by curating the dataset and proving clinical efficacy. In this project, we will focus on the technical aspects of building a segmentation model and integrating it into the clinician's workflow, leaving the dataset curation and model validation questions largely outside the scope of this project.

**What You Will Build**

In this project you will build an end-to-end Al system which features a machine learning algorithm that integrates into a clinical-grade viewer and automatically measures hippocampal volumes of new patients, as their studies are committed to the clinical imaging archive.

Fortunately you won't have to deal with full heads of patients. Our (fictional) radiology department runs a HippoCrop tool which cuts out a rectangular portion of a brain scan from every image series, making your job a bit easier, and our committed radiologists have collected and annotated a dataset of relevant volumes, and even converted them to NIFTI format!

You will use the dataset that contains the segmentations of the right hippocampus and you will use the U-Net architecture to build the segmentation model.

After that, you will proceed to integrate the model into a working clinical PACS such that it runs on every incoming study and produces a report with volume measurements.

**The Dataset**

We are using the "Hippocampus" dataset from the Medical Decathlon competition. (<http://medicaldecathlon.com/>) This dataset is stored as a collection of NIFTI files, with one file per volume, and one file per corresponding segmentation mask. The original images here are T2 MRI scans of the full brain. As noted, in this dataset we are using cropped volumes where only the region around the hippocampus has been cut out. This makes the size of our dataset quite a bit smaller, our machine learning problem a bit simpler and allows us to have reasonable training times. You should not think of it as "toy" problem, though. Algorithms that crop rectangular regions of interest are quite common in medical imaging. Segmentation is still hard.

**Local Environment**

If you would like to run the project locally, you would need a Python 3.7+ environment with the following libraries for the first two sections of the project:

• PyTorch (preferably with CUDA)

• nibabel • matplotlib

• numpy.

• pydicom

• Pillow (should be installed with pytorch)

• tensorboard

In the 3rd section of the project we will be working with three software products for emulating the clinical network. You would need to install and configure:

• Orthanc server for PACS emulation

• OHIF zero-footprint web viewer for viewing images. Note that if you deploy OHIF from its github repository, at the moment of writing the repo includes a yarn script (orthanc: up) where it downloads and runs the Orthanc server from a Docker container. If that works for you, you won't need to install Orthanc separately.

• If you are using Orthan (or other DICOMWeb server), you will need to configure OHIF to read data from your server. OHIF has instructions for this: https://docs.ohif.org/configuring/datasource.html

• In order to fully emulate the Udacity workspace, you will also need to configure Orthanc for auto-routing of studies to automatically direct them to your Al algorithm. For this you will need to take the script that you can find at section3/src/deploy\_scripts/route\_dicoms. lua and install it to Orthanc as explained on this page: https://book.orthancserver.com/users/lua.html

• DCMTK tools for testing and emulating a modality. Note that if you are running a Linux distribution, you might be able to install demtk directly from the package manager (e.g. apt-get install dcmtk in Ubuntu)

**Project Rubric:**

Part 1: Curating a Dataset of Brain MRIs

1. Dataset has been cleaned and outliers have been removed

* Correctly identified and removed the irrelevant files from the given dataset through inspection of the dataset

2. The project shows an understanding of how to apply medical metadata inspection methods to discover the physical dimensions of anatomical structures.

* The project shows an understanding of how to apply medical metadata inspection methods to discover the physical dimensions of anatomical structures.

3. The project shows an understanding of how to extract pixel data for visualization.

* Jupyter Notebook contains renderings of medical volume slices that help inspect dataset slices and validate assumptions that one might have about how pixel data is stored in the arrays read from disk.

Part 2: Training a Segmentation CNN

Expected results

Please put the artefacts from Section 2 here:

• Functional code that trains the segmentation model

• Test report with Dice scores on test set (can be json file). Your final average Dice with the default model should be around .90 • Screenshots from your Tensorboard (or other visualization engine) output, showing Train and Validation loss plots, along with images of the predictions that your model is making at different stages of training • Your trained model PyTorch parameter file (model.pth)

Suggestions for making your project stand out

• Can you write a 1-page email explaining what your algorithm is doing to a clinician who will be trying it out, but whom you never met? Make sure you include performance characteristics with some images. Try using their language and think of what would be the important information that they are looking for? • Implement additional metrics in the test report such as Jaccard score, sensitivity or specificity. Think of what additional metrics would be relevant.

• In our dataset we have labels of 2 classes - anterior and posterior segments of the hippocampus. Can you train a version of model that segments the structure as a whole, only using one class? Is the performance better, the same or worse? • Write up a short report explaining requirements for your training process (compute, memory) and suggestions for making it more efficient (model architecture, data pipeline, loss functions, data augmentation). What kind of data augmentations would NOT add value?

• What are best and worst performing volumes? Why do you think that's the case?

Machine learning scripts run without errors and perform training and validation of the machine learning model.

* There should be no <YOUR CODE HERE> blocks in the .py files of the project. All the TASK comments should be followed by blocks of code that perform the required actions or answers to questions.
* Out folder contains model.pth file, about ~100Mb in size

Project shows evidence that a system was established allowing the monitoring of progress via Tensorboard

* Script establishes proper logging of scalar and image data into Tensorboard folders, and monitoring is performed using the Tensorboard server.
* Output folder includes screenshots of train/validation loss plots.
* Plots could look like this: (ex: TensorBoard Scalars for Step)

**Create a test code that runs without errors and computes volumetric performance measurements.**

* **Code in utils/volume\_stats.py/Jaccard3D should contain no <YOUR CODE HERE BLOCK>, should contain implementation of the metric and return the computed score.**
* **Out folder contains results.json file that is a correct JSON and has at least Dice and Jaccard metrics.**

**Integrating into a Clinical Network**

The inferencing code for DICOM volumes is complete

* All TASK items in inference\_dcm.py should be addressed. A sample report file should be included along with a screenshot/png/jpg version of the said report.
* A good report screenshot may look like this:

Complete inferencing code for creating reports and pushing them back.

* The student’s report can be viewed in the OHIF image viewer solution. The report at least has numerical values of the volume of the hippocampus structure. Here is an example:

Create a validation plan.

* Out folder contains a validation plan. The plan should be in the freeform format, about 1-2 pages and should hit on topics:
  + What is the intended use of the product?
  + How was the training data collected?
  + How did you label your training data?
  + How was the training performance of the algorithm measured and how is the real-world performance going to be estimated?
  + What data will the algorithm perform well in the real world and what data it might not perform well on?

Suggestions to Make Your Project Stand Out

* Write an explanation of how the algorithm works for clinicians.
* Explain requirements for the training process (compute, memory), suggestions for making it more efficient (model architecture, data pipeline, loss functions, data augmentation). What kind of data augmentations would NOT add value?
* Implement additional metrics in testing reports - sensitivity, specificity, accuracy, etc. Include an explanation of those in the #1 writeup.
* Propose a better way of filtering study for the correct series.
* Can you think of what would make the report you generate from your inference better? What would be the relevant information that you could present which would help a clinician better reason about whether your model performed well or not? Can you make it look nicer by making it an RGB image (hint - lookup in DICOM spec(opens in a new tab))?
* Try to construct a fully valid DICOM as your model output (per DICOM PS3.3#A8(opens in a new tab)) with all relevant fields. Construction of valid DICOM has a very calming effect on the mind and body.
* Try constructing a DICOM image with your segmentation mask so that you can overlay it on the original image using the clinical image viewer.

Submission Checklist:

[ ] Everything in the Rubric is complete.

The following are in Section 1's / section1/out/ folder/directory.

[ ] Curated dataset with labels, as collection of NIFTI files.

[ ] A Python Notebook or Python File with the results of your Exploratory Data Analysis.

The following are in Section 2's / section2/out/ folder/directory.

[ ] Functional code that trains the segmentation model.

[ ] Test report with Dice scores on test set (can be json file).

[ ] Screenshots from your Tensorboard (or other visualization engine) output.

[ ] Your trained model PyTorch parameter file (model.pth)

The following are in Section 3's /section3/out/ folder/directory.

[ ] Code that runs inference on a DICOM volume and produces a DICOM report.

[ ] A report.dcm file with a sample report.

[ ] Screenshots of your report shown in the OHIF viewer.

[ ] 1-2 page Validation Plan.

Reviewer Notes:

Section 1:

1) Data Cleaning:   
It appears that you have not met the dataset curation requirement of having 260 images and 260 label files. In your script's output, you have kept 192 files (likely including both images and labels) and removed 70 files as outliers.

To meet the project specifications, your final curated dataset should include exactly 260 image files and 260 label files. Please review your script to ensure that you have correctly identified and removed only the irrelevant files while preserving the complete set of required brain MRI files. It might be helpful to re-evaluate the lower\_thresh and upper\_thresh parameters or adjust your logic for determining outliers to ensure you retain the correct number of files.

2) Histogram: It looks like the histogram you provided does not match the requirements outlined in the project instructions. The expected histogram should show a clear plot of anterior volumes with a histogram structure similar to the provided example (with appropriately set bins, axis ranges, and labeled axes).

Your current histogram uses a yellow color and covers a different volume range (2000-5000 mm³), while the example in the project uses green bars in the 1000-2600 mm³ range, clearly showing more granular distribution and volume bins that better reflect the data.

To meet the project requirements, please update your histogram to:

* Use the appropriate dataset (anterior volumes only)
* Adjust the volume range to 1000-2600 mm³ (or match the example as closely as possible)
* Set the number of bins to reflect the dataset’s distribution clearly
* Use clear axis labels and titles as shown in the example
* If you need help adjusting the plotting parameters or understanding how to select the anterior volumes only, you can post a question in the Knowledge Hub.

Please make these updates and resubmit to meet the requirements.

**\*\*Hi, I have updated the final curated dataset to include 260 images and 260 label files based on only removing the most extreme outliers. This way the dataset is consistent with prior assessments. In general, outlier definition can be determined by different methods, thus I do not think this would be a hard threshold, however it is helpful to understand what is expected. Thus, I have now updated the script in Section 1 for this purpose to ensure the previously determined number of files is consistent.**

**I have also updated the histogram to only include the anterior volumes of the hippocampus, adjusted the volume range to 1000-2600 mm³, set the number of bins to reflect the dataset distribution and also increase granularity, and used clear axis labels similar to the example image provided. Thank you for providing that image as it is helpful to see the expected distribution output.**

Section 2:

1) Python Files: It seems that your project is not fully complete. I see some sections in your Python files (like volume\_stats.py) that still contain <YOUR CODE HERE> placeholders or unfinished pass statements. According to the project requirements, there should be no <YOUR CODE HERE> or unfinished blocks in any of the .py files, and all TASK comments should be followed by fully implemented code blocks.

Please ensure that:

* All TASK blocks are completed with appropriate code to implement the intended functionality (e.g., Dice3D and Jaccard3D functions in volume\_stats.py).
* There are no placeholder tags like <YOUR CODE HERE> remaining.
* The final project contains a model.pth file in the section 2 out directory, approximately 100MB in size, which indicates a complete training run.

You will need to complete the missing implementations and re-run your scripts to ensure everything runs without errors and performs as expected.

2) Python Files It looks like the required code for the Jaccard3D function in utils/volume\_stats.py is not yet implemented. The <YOUR CODE HERE> placeholders are still present, and there is no logic provided to compute the Jaccard similarity coefficient.

According to the project requirements, you must:

Remove all <YOUR CODE HERE> placeholders and replace them with complete, working code to compute the Jaccard3D metric.

Ensure the function returns the computed Jaccard index as a float.

Confirm that the results.json file in your output folder includes the Dice and Jaccard metrics.

Please complete the implementation of Jaccard3D (and Dice3D if also incomplete) and ensure the JSON output is valid.

Code in utils/volume\_stats.py/Jaccard3D should contain no <YOUR CODE HERE BLOCK>, should contain implementation of the metric and return the computed score.

Out folder contains results.json file that is a correct JSON and has at least Dice and Jaccard metrics.

**\*\*For Section 2, I have updated the python files to ensure that the** [**VolumeStats.py**](http://volumestats.py) **script and Jaccard3D functions have the required code needed to compute the Jaccard similarity coefficient. I have ensured that the Jaccard3D metric returns the computed Jaccard index as a float, and the results.json file includes the Dice and Jaccard metrics. There are no remaining code blocks missing implementation.**

Section 3:

1) Layout of Report with DICOM volume inference step:   
It seems that you have not yet included the required sample report file for the DICOM volume inference step. According to the project requirements, you should provide:

A sample report file (e.g., PDF or text) that summarizes the results of the inference.

A screenshot or .png/.jpg version of this report showing axial slices and numerical results, similar to the example in the rubric.

The provided images demonstrate segmentations and predictions, but the final report itself is missing.

Please generate this report and include a screenshot version to demonstrate that the final inference step is complete and adheres to clinical reporting standards.

2) OHIF Viewer:

The project does not meet the requirement for integrating the code for DICOM volumes and pushing them back for report viewing in the OHIF image viewer solution. According to the rubric, the report should be viewable in OHIF with at least the numerical values of the hippocampus volume structure.

There is no evidence in your project submission that demonstrates this functionality (such as screenshots from the OHIF viewer with the appropriate measurements and overlays).

To meet this requirement, please ensure:

* Your pipeline properly integrates and pushes the final report to the OHIF viewer.
* You provide clear evidence (e.g., screenshots of the OHIF viewer interface with your data).
* The student’s report can be viewed in the OHIF image viewer solution. The report at least has numerical values of the volume of the hippocampus structure. Here is an example:

I tried to download and run the latest copy of Orthanc / OHIF viewer including with Docker, but I was not able to open it on my local machine. I have created the report nonetheless and imputed the sample overlay as an example.

Hi, for Section 1, I have updated the final curated dataset to include 260 images and 260 label files based on only removing the most extreme outliers. This way the dataset is consistent with prior assessments. In general, outlier definition can be determined by different methods, thus I do not think this would be a hard threshold, however it is helpful to understand what is expected. Thus, I have now updated the script in Section 1 for this purpose to ensure the previously determined number of files is consistent. I have also updated the histogram to only include the anterior volumes of the hippocampus, adjusted the volume range to 1000-2600 mm³, set the number of bins to reflect the dataset distribution and also increase granularity, and used clear axis labels similar to the example image provided. Thank you for providing that image as it is helpful to see the expected distribution output. For Section 2, I have updated the python files to ensure that the [VolumeStats.py](http://volumestats.py) script and Jaccard3D functions have the required code needed to compute the Jaccard similarity coefficient. I have ensured that the Jaccard3D metric returns the computed Jaccard index as a float, and the results.json file includes the Dice and Jaccard metrics. There are no remaining code blocks missing implementation. For Section 3, I completed the inference steps. However, I tried to download and run the latest copy of Orthanc / OHIF viewer including with Docker, but I was not able to open it on my local machine. I have created the report nonetheless and imputed the sample overlay as an example.

**Course 4: Introduction to AI and EHR**

Outline of Jupyter Notebook:

Udacity Jupyter Notebook Outline: EHR Data Security and Analysis

1. Dataset Schema Analysis

• Dataset Introduction: Heart Disease Dataset from UCI ML Repository

• Data Sources: 4 medical institutions (Hungarian Institute of Cardiology, University

Hospitals in Zurich and Basel, V.A. Medical Center Long Beach and Cleveland

Clinic)

• Modeling Objective: Predict incidence of heart disease

• Schema Overview: 14 attributes including age, sex, chest pain type, blood

pressure, cholesterol, etc.

• Data Loading: Import processed Cleveland Clinic dataset with column headers

2. Analyze Value Distributions

• Categorical Distribution Visualization: Functions to create bar plots for categorical

features

• Gender Distribution Analysis: Visualization of male vs female distribution

• Chest Pain Type Analysis: Distribution of different chest pain categories

• Distribution Types Review:

◦ Normal distribution examples

◦ Uniform distribution examples

◦ Right-skewed distribution identification (resting blood pressure)

3. Missing Values and Data Quality

• Numerical Feature Scaling: Analysis of min/max ranges for numerical features

(for: age, blood pressure, cholesterol, heart rate, ST depression)

• Missing Value Detection: Function to check null values and zero percentages

across all columns

• Data Quality Assessment: Percentage analysis of missing and zero values per

feature

4. Outliers

• Outlier Detection: Box plot visualizations for identifying outliers

• Age Distribution: Box plot analysis for age feature

• Cholesterol Distribution: Box plot analysis for cholesterol feature

5. High Cardinality

• Cardinality Concept: Definition and importance in EHR datasets

• High Cardinality Simulation: Function to create synthetic high-cardinality feature

(principal diagnosis codes)

• Cardinality Analysis: Count unique values for categorical features

• Comparison Table: Cardinality values ranging from 2 (binary features) to 270

(simulated diagnosis codes)

6. Demographic Analysis

• Age Binning: Convert continuous age to categorical age groups (10-year bins)

• Demographic Visualization:

◦ Age distribution across bins

◦ Gender and age distribution combined analysis

• Population Demographics: Count plots showing patient distribution by age

groups and gender

Technical Components

• Libraries Used: pandas, numpy, sklearn, seaborn, matplotlib

• Optional Tool: TensorFlow Data Validation (TFDV) for advanced EDA

• Data Processing: Categorical encoding and binning techniques

• Visualization Methods: Bar plots, histograms, box plots, count plots

This notebook provides a comprehensive exploratory data analysis framework specifically designed for healthcare data, emphasizing critical considerations like data privacy, missing values, outliers, and high cardinality features common in EHR datasets.

Videos

1.1: introduces Michael D'Andrea, Principal Data Scientist at Genentech, who shares his background and passion for applying AI to electronic health record (EHR) data in healthcare. Drawing from his experience at Genentech and Change Healthcare, he highlights the opportunities and challenges of using vast EHR datasets to improve clinical trials, reduce patient risk, and create more effective treatments. Michael emphasizes the importance of reducing bias and increasing accessibility in healthcare, with motivations rooted in fighting diseases like cancer and addressing urgent needs such as the COVID-19 pandemic.

1.2: The video "Historical Context of AI in EHR" provides an overview of the evolution from written healthcare records to electronic health records (EHR), highlighting how this transition created new opportunities and challenges for data management in healthcare. It explains that EHRs, also known as electronic medical records (EMRs), are digital versions of a patient’s healthcare documentation, with the terms EHR and EMR used interchangeably. The video notes the significance of the Health Insurance Portability and Accountability Act (HIPAA) of 1996 in facilitating this shift. It emphasizes that moving to electronic records not only required digitizing old paper data but also opened the door for artificial intelligence to enhance data usability, including areas like genomics, clinical trial analysis, and predictive diagnostics. The segment concludes by setting the stage for further discussion on the current landscape and AI’s expanding role in EHR data.

1.3: The video "Landscape of EHR Data" introduces the fundamentals of Electronic Health Records (EHRs), which are digital versions of patients’ paper charts and encompass data from doctor visits, pharmacies, dentists, radiology, and wearable devices. EHRs are crucial for personalizing healthcare, improving diagnoses, reducing errors, and enabling large-scale data analysis for drug discovery and public health insights. The video highlights the massive scale of EHR data in the U.S.—with around 30 billion healthcare claims annually and healthcare costs nearing $3.24 trillion in 2019, about 15% of the U.S. GDP. It also explains that EHR data comes from diverse sources, providing a comprehensive, longitudinal view of patient health. AI and EHR integration offers opportunities to increase healthcare access, reduce administrative errors, enable early disease prediction, and develop more precise treatments by combining EHRs with genetic and imaging data. Despite technological advances, access to healthcare remains a global issue, with nearly half the world lacking essential services and 45% of U.S. adults inadequately insured. The video concludes by encouraging the use of AI and EHR data to drive innovation and improve healthcare delivery.

1.4: The "Applying AI to EHR Data" course preview introduces learners to the essential topics they will cover, starting with EHR data security and analysis, emphasizing the importance of safeguarding patient information and understanding relevant global regulations. The course follows the CRISP-DM framework for exploratory data analysis to ensure datasets are representative and ready for modeling. Learners will explore medical code sets such as ICD-10, CPT, NDC, and RxNorm, and learn how to group codes for machine learning tasks. The curriculum then covers EHR data transformation, feature engineering with TensorFlow, and choosing appropriate data levels (line, encounter, or longitudinal). Finally, students will build, evaluate, and interpret models, focusing on best practices, uncertainty and bias assessment, demographic bias mitigation using tools like Aequitas, Bayesian probability, and model interpretability with Shapley values. By the end, participants gain practical skills for leveraging AI in healthcare.

1.5: The video covers the relevant tools for working with AI and Electronic Health Records (EHR). It emphasizes the use of Jupyter Notebooks and various Python libraries such as NumPy, Pandas, Seaborn, Matplotlib, TensorFlow, TensorFlow Probability, Aequitas, and Shapley for data analysis and machine learning tasks. The speaker highlights the importance of having intermediate Python skills and a basic understanding of machine learning to succeed in the course. Additionally, the video discusses the need for cloud computing resources from providers like Google Cloud Platform, Amazon AWS, and Azure for handling large datasets. Other programming languages and tools relevant to AI in healthcare, such as R, Julia, SQL, PyTorch, and Scikit-learn, are also mentioned. Finally, the video touches on EHR systems like Cerner and Epic, noting that while students may not work directly with these systems, it's beneficial to be aware of them.

Additional Resources

• Python

• Numpy.

• Pandas

• Seaborn

• MatPlotLib

• TensorFlow 2.0/2.1

• TensorFlow Feature Columns • TensorFlow Probability.

• TensorFlow DenseFeatures • Shapley.

• Aequitas

• Google Cloud Platform

• Amazon AWS

• Microsoft Azure

• R

• Julia

• SQL

• Big Query.

• AWS Athena

• Scikit Learn

• PySpark

• Cerner

• Epic

1.6: The video outlines the project for the course, where you will act as a data scientist for a healthcare startup that has developed a new diabetes drug ready for clinical trials. Your tasks include building a predictive model to identify suitable patient types for testing the drug, creating a regression model to estimate hospitalization time for patients, and conducting a demographic bias analysis to ensure fairness in your model. The video emphasizes the use of the UCI diabetes readmission dataset and highlights the learning objectives, such as using healthcare codes, managing datasets, and accounting for bias and uncertainty in your models. The speaker expresses excitement about the project and encourages students to engage with the material.

Part 2: Data Security & Privacy, Exploratory Data Analysis, Demographic Dataset Analysis

2.1: The video provides an overview of the key concepts related to Electronic Health Records (EHR) data security and analysis. It begins by discussing the importance of data security and privacy in healthcare, highlighting key standards and regulations that govern the handling of EHR data. The video then transitions into exploratory data analysis, emphasizing the need to understand datasets through schema analysis, value distributions, missing values, and the cardinality of categorical features. Additionally, the video covers demographic dataset analysis, which helps assess how representative a dataset is before training machine learning models. By the end of the lesson, viewers will be equipped with essential knowledge to work effectively with healthcare datasets while ensuring the integrity and security of patient information.

2.2: The video discusses the importance of data privacy and security in the context of Electronic Health Records (EHR). It highlights that protecting personal health information is crucial, as breaches can lead to identity theft and discrimination against patients with certain medical conditions. The video emphasizes the need for healthcare professionals to be aware of regulations and best practices to ensure compliance with privacy laws, such as HIPAA. It also provides examples of data breaches and their consequences, illustrating the risks associated with mishandling sensitive information. The video concludes by stressing the significance of encryption and proper handling of Protected Health Information (PHI) to safeguard patient data and maintain trust in the healthcare system.

2.3: Key Healthcare Data Security and Privacy Standards: The video discusses key healthcare data security and privacy standards, focusing on regulations like HIPAA and HITECH in the U.S. It explains the importance of understanding these regulations for anyone working with electronic health records (EHR). The video highlights that HIPAA protects patient information and sets standards for data privacy and security, while HITECH updates HIPAA to address technological advancements. It also briefly mentions international regulations like GDPR in the EU and DPA in the UK, emphasizing the need for familiarity with these laws to ensure compliance and protect patient data. The video concludes by stressing the significance of these standards in the context of healthcare data analysis and AI applications.

2.4: Storing and Accessing PHI Data: The video discusses the importance of storing and accessing Protected Health Information (PHI) data securely. It emphasizes that while encryption is not strictly required under the Security Rule, it is considered a best practice to protect sensitive data. The video outlines the need for HIPAA-compliant cloud providers and highlights the minimum necessary standard, which mandates limiting access to PHI to only what is necessary for a specific purpose. Additionally, the video explains the process of de-identifying datasets to reduce privacy risks, detailing two methods: expert determination and safe harbor, which involves removing specific identifiers. The speaker stresses the importance of consulting legal and compliance teams when handling PHI and the potential legal risks associated with re-identification. Overall, the video underscores the critical nature of data security in healthcare settings.

HIPAA (Health Insurance Portability and Accountability Act) regulations are designed to protect the privacy and security of individuals’ protected health information (PHI). HIPAA allows covered entities (such as healthcare providers, insurers, and their business associates) to use and disclose PHI for treatment, payment, and healthcare operations, as well as in certain other situations required by law or with patient authorization. However, HIPAA does not permit the use or disclosure of PHI for unauthorized purposes, such as marketing or sharing with unauthorized individuals or organizations, and it requires that only the minimum necessary information is accessed or shared to accomplish the intended purpose.

In day-to-day practice, organizations and providers must ensure that PHI is secured both at rest and in transit, commonly through encryption and access controls. They need to use only HIPAA-compliant services, verify business associate agreements (BAAs), and follow the minimum necessary standard by restricting PHI access to only those who need it for their job functions. Staff must be trained on privacy practices, and any de-identification of data must follow approved HIPAA methods—either via expert determination or the Safe Harbor approach. Compliance teams should be consulted for any use cases involving PHI or de-identified data to minimize legal risk and ensure all data handling processes meet HIPAA requirements.

2.5: PII Data Solution:

Key takeaways

* Be aware of what you do not know
* Your organization likely has compliance protocols and rules
* There is required HIPPA training for those is the US

Personal Learning

* privacy is evolving
* Industrial methods are gaining more acceptance across different fields
* Healthcare methods are also gaining acceptance

2.6 Importance of EDA (exploratory data analysis): The video discusses the importance of exploratory data analysis (EDA) in the context of healthcare data. It emphasizes that understanding the data before building models is crucial to avoid common pitfalls, such as misrepresenting data or overlooking null values. The video introduces the CRISP-DM framework, highlighting that EDA falls under the data understanding phase. Key reasons for conducting EDA include discovering features that may lead to data leakage, defining modeling objectives and evaluation metrics, and informing strategies for handling missing data. The speaker also notes that EDA can help identify subsets of features for modeling and feature engineering. Overall, the video underscores the critical role of EDA in ensuring the accuracy and effectiveness of data science projects in healthcare.

In EHR modeling, data leakage can take many forms beyond temporal leakage. Common types include using post-outcome data, derived features that encode the target (e.g., diagnosis codes), duplicate patient records across train-test splits, and improperly engineered features that include future information. Metadata such as timestamps or provider IDs can also leak outcome-related signals. Label leakage occurs when features implicitly contain the outcome, and cohort selection bias arises when models are trained on pre-filtered patient groups. Preventing these requires careful EDA, timeline validation, and clinically informed feature selection to ensure model inputs reflect only information available at the prediction time.

2.7 Dataset Schema Analysis: The video discusses the importance of understanding the schema of a dataset before analyzing its statistical properties. It introduces a heart disease dataset from the University of California Irvine, which contains 76 attributes, but only 14 will be used for analysis in the course. The speaker emphasizes the need to review the schema, identify predictor attributes, and distinguish between categorical and continuous features. Examples are provided, such as sex as a categorical feature and age as a continuous feature. The video highlights the significance of gaining a solid understanding of the schema and collaborating with domain experts to ensure effective analysis of EHR data.

Key Things to Consider:

• Identify the predictor

• Identify categorical, numerical features

• Work with SMEs and Domain Experts

• Domain knowledge is key to representing data correctly

You can find this dataset in the notebook on this page to inspect the dataset as well as use the link below to get more specific information for the different categories.

<https://archive.ics.uci.edu/dataset/45/heart+disease>

2.8: Value Distributions: Normal, Uniform, Skewed, Binomial, etc

2.9: Missing Values and Outliers:

The video on "Missing Values and Outliers" discusses the significance of identifying and handling missing data in datasets, particularly in healthcare. It explains three classifications of missing data:

1. \*\*MCAR (Missing Completely at Random)\*\*: Data is missing without any systematic reason.

2. \*\*MAR (Missing at Random)\*\*: The missingness is related to other observed data but not the missing data itself.

3. \*\*MNAR (Missing Not at Random)\*\*: The missingness is related to the value of the missing data itself.

1. MCAR (Missing Completely at Random):

• Example: A researcher is conducting a survey and accidentally skips a few questions due to a technical glitch in the survey software. The missing responses are entirely random and unrelated to the participants' characteristics or the content of the survey. In this case, the missing data does not introduce any bias because it is not systematically related to any other variables.

2. MAR (Missing at Random):

• Example: In a clinical study, older participants are less likely to report their income compared to younger participants. The missing income data is related to the age of the participants but not to the income itself. Here, the missingness can be explained by other observed variables (age), making it MAR. If we account for age in our analysis, we can still make valid inferences about the data.

3. MNAR (Missing Not at Random):

• Example: In a study on mental health, individuals with severe depression may be less likely to respond to questions about their condition due to their mental state. The missing data (responses from those with severe depression) is related to the value of the missing data itself (the severity of their depression). This creates a situation where the missingness is not random and can lead to biased results if not properly addressed.

In summary, the key difference is that MAR can be explained by other observed data, while MNAR cannot. If both examples seem to depend on an individual metric, the distinction lies in whether that metric can be explained by other observed variables (MAR) or if it is inherently related to the missing values themselves (MNAR). If you have further questions or need clarification, feel free to ask!

The video emphasizes the importance of understanding the nature of missing values to inform appropriate imputation strategies. It also discusses the impact of outliers on data analysis, illustrating how they can skew results and affect model performance. Techniques for detecting and handling outliers, such as using box plots and the IQR method, are briefly mentioned. Overall, the video stresses the need for careful analysis of missing values and outliers to ensure the integrity of data-driven decisions in healthcare.

MCAR: missing completely at random (no systemic reason for the missing data, ex: instrumentation issue)

MAR: missing at random; systematic relationship between data and probability of missing data.. Ex: surveys that more likely to reveal some pieces of data

MNAR: is missing not at random; relationship between a value in the dataset and the missing values.

Approaches: removal (values outside 1.5IQR range, vs. imputing a statistical value

Function to check the percent of missing and zero values in data set:

def check\_for\_missing\_and\_null(df):

null\_df = pd.DataFrame({'columns': df.columns,

'percent\_null': df.isnull().sum() \* 100 / len(df),

'percent\_zero': df.isin([0]).sum() \* 100 / len(df)

} )

return null\_df

2.10: Analyzing Dataset for High Cardinality: categorical variable with many dimensions (ex: diagnostic codes, requires dimensionality reduction for modeling).

High Cardinality

Cardinality: refers to the number of unique values that a feature has and is relevant to EHR datasets because there are code sets such as diagnosis codes in the order of tens of thousands of unique codes. This only applies to categorical features and the reason this is a problem is that it can increase dimensionality and makes training models much more difficult and time-consuming.

How do we define a field with high cardinality?

• Determine if it is a categorical feature.

• Determine if it has a high number of unique values. This can be a bit subjective but we can probably agree that for a field with 2 unique values would not have high cardinality whereas a field like diagnosis codes might have tens of thousands of unique values would have high cardinality.

• Use the nunique () method to return the number of unique values for the categorical categories above.

2.13: Demographic Analysis: The video on "Demographic Analysis" emphasizes the importance of ensuring that datasets used in healthcare are representative of the general population. It discusses how demographic analysis helps identify potential biases in data, which is crucial for developing fair and effective healthcare solutions. The speaker explains that when conducting demographic analysis, it's essential to group data into categories, such as age and gender, to better understand the distribution of the population being studied.The video also highlights the use of tools and methods for analyzing demographic data, such as creating age buckets and examining the distribution of different demographic groups within the dataset. By doing so, data scientists can assess whether their models will perform well across diverse populations and identify any areas that may require balancing or additional representation. Overall, the video underscores the critical role of demographic analysis in ensuring equitable healthcare outcomes.

Part 3: EHR Code Sets

3.1 EHR Code Sets Overview:

The "EHR Code Sets Overview" video introduces the concept of EHR code sets and their significance in navigating electronic health record data. The speaker explains that code sets are standardized codes representing various medical concepts, such as diagnoses and procedures, which facilitate data comparison across different EHR systems.

The video outlines the key components that will be covered in the lesson, including:

1. \*\*Diagnosis Codes\*\*: Focus on ICD10-CM, including the transition from ICD9 to ICD10.

2. \*\*Procedure Codes\*\*: Discussion of ICD10-PCS, CPT, and HCPCS codes.

3. \*\*Medication Codes\*\*: Overview of NDC Directory and RxNorm code sets.

4. \*\*Grouping and Categorization\*\*: Introduction to CCS for grouping procedure codes.

By the end of the lesson, viewers will understand how to read and use these codes effectively, as well as the challenges associated with working with medication code sets. The speaker encourages viewers to prepare for a detailed exploration of the various acronyms and codes in healthcare data.

3.2 Codes Set Background:

The video titled "Codes Set Background" provides an introduction to code sets used in Electronic Health Records (EHR). It explains that code sets are standardized codes linked to accepted coding standards, such as ICD-10 and CPT codes, which are essential for encoding medical diagnoses, procedures, and other healthcare-related information. The speaker emphasizes the importance of having a standardized way to represent common diagnoses, medications, procedures, and lab tests to facilitate data sharing and analysis across different healthcare providers. The video also highlights the role of medical coders in converting unstructured medical information into structured codes that can be used in EHR systems. Additionally, the video touches on the context of medical encounters, differentiating between inpatient and outpatient settings, and the significance of working with subject matter experts to understand the intricacies of healthcare coding. Overall, the video sets the stage for a deeper exploration of specific code sets and their applications in healthcare data analysis.

3.3 Diagnosis Codes Part 1:

The "Diagnosis Codes Part 1" video explains the critical role of diagnosis codes in healthcare, particularly within Electronic Health Records (EHRs). Diagnosis codes are essential for documenting patient conditions and facilitating standardized communication among healthcare providers and health systems. The video focuses on the ICD-10-CM system, which is the U.S. clinical modification of the World Health Organization’s ICD-10 standard, maintained by the CDC. ICD-10-CM is used not only for medical billing but also for tracking disease epidemiology and supporting public health initiatives. The transition from ICD-9 to ICD-10 in the U.S. in 2014 marked a significant advancement, expanding the number of codes from about 14,000 to nearly 70,000 and increasing the maximum code length to seven characters. This change allowed for much greater detail and specificity in recording diagnoses, including factors like laterality and episode of care. ICD-10-CM codes are structured in three parts: the first three characters indicate the diagnostic category, the next set of up to three characters detail etiology, anatomic site, or manifestation, and the final character (the extension) often specifies the episode of care, especially for injuries. Understanding this structure is foundational for anyone working with EHR data, as the granularity and standardization provided by ICD-10-CM support accurate patient record-keeping, billing, clinical research, and analytics. The video sets the stage for deeper exploration of diagnosis codes and their impact on healthcare data management.

3.4 Diagnosis Codes Part 2: In "Diagnosis Codes Part 2," the video explains key concepts related to the assignment and prioritization of diagnosis codes within Electronic Health Records (EHR). It highlights that a patient may require several encounters before a diagnosis—and its corresponding code—can be confirmed, and diagnosis codes should not be repeated within a single encounter, but often appear across multiple encounters as ongoing conditions are treated. The video distinguishes between **primary, principal, and secondary diagnosis codes:** the primary code reflects the condition consuming the most resources, the principal code is determined after hospitalization as the chief cause, and secondary codes document coexisting conditions. Examples clarify these roles, such as a patient with knee replacement surgery (primary code) and pre-existing diabetes (secondary code). The lesson also provides resources for further ICD10-CM code exploration and stresses the importance of understanding code prioritization for accurate clinical documentation.

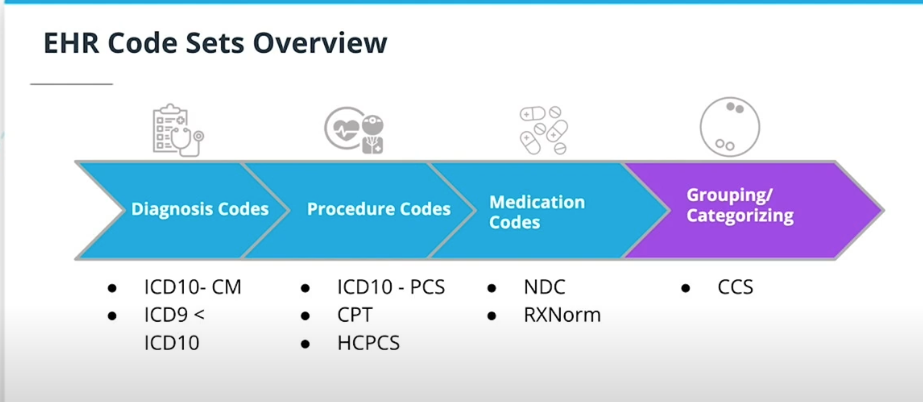
3.8 Procedure Codes:

ICD-10-PCS (International Classification of Diseases, 10th Revision, Procedure Coding System) is specifically designed for coding inpatient hospital procedures in the United States. Each ICD-10-PCS code consists of exactly seven alphanumeric characters, with each character representing a distinct aspect of the procedure: section, body system, root operation, body part, approach, device, and qualifier. There are no decimals or punctuation, and each position in the code has a standardized meaning, allowing for high specificity and consistency. The structure enables over 70,000 unique codes, supporting detailed clinical documentation and comprehensive data analysis for hospital-based procedures.

CPT (Current Procedural Terminology) codes are structured as five-digit numeric codes (e.g., 99213), with the possibility of two-character modifiers (e.g., -25, -59) appended to provide additional information about the procedure or service, such as location, laterality, or distinct procedural circumstances. CPT codes are divided into three categories: Category I (most commonly used, covering established medical procedures and services), Category II (optional tracking codes for performance measurement), and Category III (temporary codes for emerging technologies or services). The codes are maintained and updated annually by the American Medical Association, ensuring they reflect current clinical practice and technological advances.

HCPCS (Healthcare Common Procedure Coding System) is divided into two levels. Level I is identical to the CPT codes (five-digit numeric codes), while Level II codes are used for items and services not covered by CPT, such as medical equipment, supplies, and certain drugs. Level II HCPCS codes are alphanumeric, beginning with a single letter (A-V) followed by four digits (e.g., A0428 for ambulance service). These codes may also have additional modifiers for further specificity, such as site or type of service. HCPCS codes are updated annually by the Centers for Medicare & Medicaid Services (CMS) and are essential for Medicare, Medicaid, and many private payer claims, ensuring comprehensive and standardized reporting of a wide range of healthcare products and services.

3.11 Medication Codes: The "Medication Codes" video provides a comprehensive overview of how medications are classified and tracked within healthcare data systems, focusing primarily on the National Drug Code (NDC) system used in the United States since 1972. The NDC is a 10- to 11-digit code maintained by the FDA and is divided into three segments: the labeler code (identifying the manufacturer), the product code (detailing the specific drug, its name, dosage, and formulation), and the package code (indicating the form and size of the medication package). The video illustrates these segments using real-world examples, such as mapping proprietary and non-proprietary drug names and specifying routes of administration and packaging details. It also discusses the concept of crosswalks, which connect NDC codes to other code sets like HCPCS, facilitating interoperability and billing for drugs, especially injectables. A key challenge highlighted is the complexity of grouping or normalizing NDC codes for drugs with multiple formulations or combined products (e.g., acetaminophen found in many combinations), which complicates analysis and reporting. To address this, the video introduces RXNorm, a standardized naming system developed by the NIH to normalize and group medications for clearer communication across diverse healthcare systems. Overall, the video underscores the vital role of standardized medication codes in ensuring accurate tracking, billing, safety, and effective data analysis in healthcare. NDC Drug Codes List: <https://ndclist.com/search>



3.12 Grouping / Categorizing Systems: This section details how large sets of clinical codes, such as the 77,000+ ICD-10-PCS codes, are organized into meaningful categories to facilitate analysis and reduce dimensionality in electronic health record (EHR) data. The primary tool for this is Clinical Classifications Software (CCS), developed by HCUP as a government-industry partnership. CCS maps diagnosis and procedure codes from ICD code sets into single-level (285 diagnosis, 231 procedure mutually exclusive categories) or multi-level hierarchies (up to 4 levels for diagnoses, 3 for procedures), enabling both high-level grouping (e.g., cardiovascular operations) and granular sub-categorization (e.g., hypertensive heart disease). The CCS ICD-10-PCS Category Mapping File, available as a downloadable CSV, provides the mappings and category labels necessary for programmatic grouping, greatly simplifying the task for analysts and clinicians. The lesson also references other systems like MS-DRG (used for Medicare payment grouping) and SNOMED CT (for interoperability), though the course focuses on CCS. Practical exercises ask learners to use string-matching techniques to select relevant cases for conditions like colon cancer using either single-level or multi-level CCS codes. The module emphasizes both the technical and practical value of these groupers for risk adjustment, ranking, and scalable EHR data analytics, and provides links to official resources for deeper exploration.

**Summary:** ICD-10-CM diagnosis codes are not directly linked to procedure or medication codes but work to complement each other within healthcare encounters to provide a complete clinical picture. ICD-10-CM captures what conditions patients have, while procedure codes (ICD-10-PCS, CPT, HCPCS) document what was done, and medication codes (NDC, RxNorm) track prescribed drugs. The Clinical Classifications Software (CCS) encompasses both diagnosis and procedure codes, not just procedures: it maps ICD-10-CM diagnosis codes and ICD-10-PCS procedure codes into clinically meaningful categories to reduce complexity for analysis. These code sets function together rather than hierarchically, with advanced approaches like Graph Convolutional Transformers leveraging the relationships between diagnosis, procedure, and medication codes to understand clinical patterns and extract insights across patient encounters and longitudinal care histories.

**Part 4: EHR Transformations & Feature Engineering**

4.1: Transformations and Feature Engineering Overview

This lesson is divided into 3 parts:

EHR Dataset Levels: In this part, there are three levels - line, encounter, and longitudinal. By the end of this section, you will be able to identify the level of your dataset as well as conduct tests and transform your data.

Dataset Splitting Without Data Leakage: In this part, you will learn about dataset splitting without Data leakage, which can be a major issue in EHR datasets. By the end of part two, you will be able to implement some basic tests to help prevent issues when splitting data.

Feature Engineering with Tensorflow: Finally, we will cover Feature Engineering with Tensorflow. In this part, we will cover ETL (Extract, Transform, Load) using TensorFlow. This will allow you to scalably process and transform your data for modeling. You will also be able to transform datasets using the TF Feature Column API for both numerical and categorical features. The Feature Column APl can be extremely useful for transforming datasets at scale and building some unique feature types.

4.2 EHR Dataset Levels:

# Understanding EHR Dataset Levels: A Critical Foundation for Healthcare AI

Electronic Health Record (EHR) datasets are organized into three distinct hierarchical levels that form the backbone of healthcare data analysis: line level, encounter level, and longitudinal level. Understanding these levels is fundamental for healthcare data scientists and AI practitioners, as selecting the wrong level can lead to significant modeling errors and faulty assumptions that may not become apparent until deployment phases of machine learning projects.

The line level represents the most granular form of EHR data, containing a denormalized or disaggregated representation of all individual events that occur during a medical visit. Each line corresponds to a specific medical code, whether it be a diagnosis, medication prescription, laboratory test, or procedure. For example, a patient visiting a doctor for bronchitis would generate multiple line-level entries: a diagnosis code for bronchitis, a medication code for a prescribed cough suppressant, and a procedure code for any diagnostic tests performed. This level captures every discrete medical action or finding as a separate data point.

The encounter level, also known as the visit level, aggregates all line-level information from a single medical encounter into a consolidated representation. This aggregation can take the form of a single row containing arrays or lists of all relevant codes from that visit. Using the bronchitis example, the encounter level would combine the diagnosis code, medication code, and procedure code into one comprehensive record representing the entire visit. This level provides a complete picture of what transpired during a specific healthcare interaction while maintaining the encounter as the unit of analysis.

The longitudinal level, or patient level, represents the highest level of aggregation, encompassing a patient's complete medical history across multiple encounters over time. This level enables healthcare professionals and researchers to identify patterns, trends, and relationships that emerge from the culmination of various visits and treatments. In the bronchitis scenario, if a patient experiences recurring bronchitis episodes over several years, the longitudinal view might reveal insights into potential autoimmune conditions or help establish optimal treatment protocols based on the patient's historical response patterns.

Distinguishing between these levels is crucial for proper data analysis and can be accomplished through simple calculations. By comparing the total number of rows in a dataset to the number of unique encounters, analysts can determine the current level of their data. If the total rows exceed unique encounters, the data is at the line level. If these numbers are equal, the data is at the encounter level. For longitudinal data, the total rows should equal the number of unique patients, as encounter information is typically aggregated or the encounter identifier may be absent entirely.

The importance of correctly identifying and working with the appropriate EHR level cannot be overstated, as errors in level identification can lead to serious analytical problems. A common mistake involves treating line-level data as encounter-level data, which can result in severe data duplication issues. For instance, if a single encounter generates fifty line entries, incorrectly treating this as fifty separate encounters would artificially upsample certain values and introduce substantial noise into the dataset. Additionally, improper encounter selection from patient records can significantly impact model performance, particularly when specific temporal requirements exist, such as needing the earliest or latest visit for predictive modeling purposes. These issues underscore why proper dataset level identification serves as a critical first step in any EHR-based machine learning project.

4.2 Encounter Representation:

The "Encounter Representation" video explains how to transform healthcare data from a line-level format to an encounter-level format. It begins by defining what an encounter is, which refers to an interaction between a patient and a healthcare provider for the purpose of delivering care.The video illustrates the process of aggregating data by grouping it based on key identifiers such as encounter ID and patient ID. It discusses the importance of creating a synthetic dataset to demonstrate this transformation while ensuring that sensitive patient information is protected.The speaker walks through the steps of using Python and the Pandas library to perform the necessary data manipulation, including grouping and aggregating data to create a more structured encounter-level dataset. This transformation is crucial for analyzing healthcare data effectively, as it allows for a clearer understanding of patient interactions and treatments over time. Overall, the video emphasizes the significance of encounter representation in healthcare analytics

4.4-4.5: Converting EHR Data to Longitudinal Level

Converting Electronic Health Record (EHR) data from encounter level to longitudinal level involves aggregating multiple patient encounters into a comprehensive patient-centered view that captures the complete medical history over time. This transformation process begins by inspecting the encounter-level dataset, which contains individual healthcare visits with associated diagnosis and procedure codes. The conversion utilizes Pandas grouping and aggregation methods, specifically grouping data by patient ID rather than encounter ID, which consolidates all encounters for each patient into single records. During this transformation, principal diagnosis codes typically remain as single values per encounter since each visit has one primary diagnosis, while procedure codes become arrays or lists containing all procedures performed across multiple encounters. For example, a patient with five encounters might have procedure codes displayed as nested arrays showing the progression of treatments over time, with some encounters potentially having null values when no procedures were performed. This longitudinal representation is crucial for healthcare AI applications as it provides a complete temporal view of patient care, enabling better pattern recognition and predictive modeling by capturing the full spectrum of a patient's medical journey rather than isolated healthcare events.

4.8 Dataset Splitting Without Data Leakage in EHR Machine Learning

Dataset splitting is a critical yet frequently mishandled process in Electronic Health Record (EHR) machine learning that can invalidate model results if done incorrectly. The primary challenge is preventing data leakage, which occurs when information inadvertently flows between training and testing datasets, causing models to perform excellently during development but fail catastrophically in production. In longitudinal EHR datasets, this commonly happens when different encounters from the same patient are distributed across multiple splits, essentially providing the model with partial answers during training. Representative splitting is equally important, ensuring that label distributions and demographic characteristics in each partition accurately reflect real-world populations to avoid biased models. For example, having only female patients in training data and male patients in testing would create gender-specific biases that compromise model generalization. To validate proper dataset splitting, practitioners should verify that no single patient's data appears in multiple partitions, confirm that the total number of unique patients across all splits equals the original dataset's patient count, and ensure that the sum of rows across all partitions matches the original dataset length, preventing data loss during the splitting process.

4.9 How to Split Dataset at Patient Level:

This Udacity lesson video demonstrates the practical implementation of patient-level dataset splitting for EHR (Electronic Health Record) data to prevent data leakage in machine learning models. The instructor walks through a Jupyter notebook example showing how to properly partition a dataset into training and testing sets by ensuring that individual patients' data remains entirely within one partition. The process begins by extracting unique patient IDs and using a splitter function to create an 80-20 train-test split. The video emphasizes the critical importance of validation through three essential tests: first, verifying that no patient ID appears in multiple partitions (zero intersection); second, confirming that the total number of unique patients across all partitions equals the original dataset's patient count; and third, ensuring that the sum of rows across all splits matches the original dataset's total row count. The demonstration shows all three validation tests passing successfully, indicating proper dataset splitting without patient ID leakage. The instructor notes that while these tests provide confidence in the splitting methodology, additional validation may be necessary depending on specific use cases. This approach is fundamental in healthcare AI applications where patient-level data integrity must be maintained to ensure model reliability and prevent overfitting that could occur if the same patient's data appears in both training and testing sets.

Patient-level dataset splitting becomes even more critical when dealing with longitudinal EHR data, where patients may have multiple encounters, admissions, or visits spanning months or years. In these scenarios, traditional random splitting could inadvertently place a patient's early encounters in the training set while their later encounters end up in the test set, creating temporal data leakage that artificially inflates model performance. This is particularly problematic in predictive modeling for chronic diseases, readmission risk assessment, or treatment outcome prediction, where the model might inadvertently learn patient-specific patterns rather than generalizable clinical insights. Healthcare AI practitioners must also consider stratified splitting approaches that maintain the distribution of key demographic variables, comorbidities, or outcome classes across partitions to ensure representative training and testing cohorts. Additionally, when working with rare diseases or specialized populations, careful attention must be paid to maintaining sufficient sample sizes in each partition while preserving statistical power. The validation framework presented in the video should be extended to include checks for temporal consistency, ensuring that prediction timeframes don't overlap between training and testing data, and verification that critical subgroups are adequately represented across all partitions to avoid model bias that could disproportionately affect certain patient populations.

Yes, with longitudinal EHR data, all encounters for a given patient must be included in either the training or testing partition—never split across both. This patient-level splitting ensures that the model cannot learn patient-specific patterns from training data and then be evaluated on the same patient's future encounters. Several sophisticated solutions address the challenges of longitudinal data splitting while maintaining robust model evaluation. Temporal holdout splitting creates training sets using data from earlier time periods (e.g., 2018-2020) and testing sets from later periods (2021-2022), simulating real-world deployment scenarios where models predict future outcomes. Stratified patient-level splitting maintains the distribution of key variables by first grouping patients by important characteristics (age groups, primary diagnoses, severity scores) before randomly sampling within each stratum. For rare diseases or small cohorts, k-fold cross-validation at the patient level can maximize data utilization while preventing leakage—each fold contains complete patient records, and validation metrics are averaged across folds. Rolling window validation is particularly valuable for time-series predictions, where training windows slide forward chronologically, always maintaining temporal separation between training and testing periods. Advanced approaches include propensity score matching to ensure balanced cohorts across splits, especially important when dealing with selection bias in EHR data. Some practitioners implement hierarchical splitting for multi-site studies, first splitting by hospital or clinic, then by patient within each site, to account for institutional practice variations. Additionally, synthetic data augmentation techniques can help address sample size limitations while preserving patient privacy and maintaining the statistical properties necessary for robust model development and evaluation.

Not all approaches are equally suitable for longitudinal data—the choice depends on the specific research question and data characteristics. **Temporal holdout splitting** is the gold standard for longitudinal EHR data as it naturally respects the temporal ordering and simulates real-world deployment where models predict future events based on historical data. **Rolling window validation** is also highly appropriate for longitudinal studies, particularly for time-series forecasting or when evaluating model performance across different time periods.

**Stratified patient-level splitting** can be used with longitudinal data but requires careful consideration—stratification variables should be time-invariant (like baseline demographics) rather than time-varying clinical measures that could introduce bias. **K-fold cross-validation at the patient level** is less ideal for longitudinal data because it doesn't preserve temporal relationships and may place earlier and later time periods in the same fold, potentially causing subtle temporal leakage.

**Hierarchical splitting** works well when longitudinal data comes from multiple institutions or when accounting for site-specific temporal trends. **Propensity score matching** can be valuable but should be based on baseline characteristics rather than longitudinal features to avoid introducing bias.

The key principle is that any splitting approach for longitudinal data must maintain temporal integrity—training data should always precede testing data chronologically, or if using cross-validation, complete patient timelines must remain intact within each fold. Most practitioners favor temporal holdout or rolling window approaches for longitudinal EHR studies because they best reflect the real-world scenario where models trained on historical data are used to make predictions about future patients or events.

4.11 Data Splitting Exercise + Solution:

This Udacity lesson video explains the solution to a data splitting exercise for Electronic Health Record (EHR) datasets in healthcare AI applications. The instructor emphasizes a critical step that is often overlooked: identifying the correct dataset level before performing the split. The exercise demonstrates that the dataset is initially at the line level, which must be converted to the encounter level before splitting into training and test partitions to prevent data leakage. This conversion step is essential because improper dataset splitting can lead to information leakage between training and test sets, compromising model validation. The instructor reviews the same splitting methodology and validation tests covered in previous lessons, reinforcing the importance of understanding EHR dataset levels (line, encounter, and longitudinal) and their proper transformations. The solution serves as both a practical implementation guide and a conceptual review, ensuring students remember the fundamental EHR level concepts that are crucial for proper data preprocessing in healthcare machine learning applications. The video concludes by encouraging students to not only perform the split correctly but also validate their results through appropriate testing methods.

4.12 ETL with TensorFlow Dataset API

Based on the content from the Udacity lesson on "ETL with TensorFlow Dataset API," this educational video covers essential concepts for processing Electronic Health Record (EHR) data using TensorFlow's Dataset API. The lesson is part of Udacity's AI for Healthcare nanodegree program, specifically focusing on applying AI to EHR data transformations and feature engineering.

The video explains that TensorFlow's Dataset API (tf.data.Dataset) provides three key advantages for ETL processes: it processes input data in a distributed format, enables batching and parallel processing on GPUs/TPUs, and automatically builds iterators that batch process data while preventing memory issues. The instructor demonstrates a practical example using the Swiss heart disease dataset, showing how to convert a Pandas DataFrame loaded from CSV into a TensorFlow dataset. A critical limitation highlighted is that the TensorFlow Dataset API cannot handle mixed data types, requiring preprocessing to remove or convert null values. The lesson includes hands-on coding demonstrations where students learn to separate features from labels, shuffle data, and create batches for iterative loading. This approach optimizes input pipelines for distributed training and leverages TensorFlow's computational power for healthcare AI applications, making it particularly valuable for processing large-scale EHR datasets efficiently.

4.13 Numerical Features & Feature Column API

This educational content focuses on transforming and preprocessing numerical features using TensorFlow's Feature Column API for healthcare data applications. The lesson demonstrates how TensorFlow's Feature Column API simplifies data preprocessing compared to traditional tools like Scikit-learn or PySpark by abstracting normalization tasks and enabling advanced features like cross features and shared embeddings. The core workflow involves five key steps: identifying numerical feature fields, loading datasets using TensorFlow's Dataset API, creating custom normalizer functions (such as z-score normalization), applying the numeric\_column feature with the normalizer function, and leveraging the API's automated processing capabilities. The practical demonstration uses a Swiss dataset to calculate summary statistics (mean and standard deviation) for an age field, then applies z-score normalization through a custom function integrated with TensorFlow's numeric column feature. The lesson emphasizes the API's flexibility in accepting various normalization methods while maintaining scalability for large healthcare datasets. This approach is particularly valuable for EHR (Electronic Health Record) data preprocessing, where numerical features require careful normalization before model training, and the TensorFlow ecosystem provides streamlined tools for handling complex healthcare data transformations efficiently.

When comparing normalization methods for healthcare data, TensorFlow Feature Columns offer production advantages through seamless integration and scalability, but aren't necessarily superior for all use cases. MinMaxScaler scales data to [0,1] range but is highly sensitive to outliers, making it problematic for clinical data with extreme values like lab results. StandardScaler (Z-score) handles outliers better but assumes normal distributions. For healthcare applications, RobustScaler (from sci-kit learn) emerges as the optimal choice, using median and IQR instead of mean/standard deviation, making it resistant to the clinically meaningful outliers common in EHR data. QuantileTransformer also excels with highly skewed biomarkers by mapping to uniform distributions. The ideal approach combines the best of both worlds: implementing RobustScaler logic within TensorFlow Feature Columns using custom normalizer functions, providing both statistical robustness for healthcare data and production scalability. While TensorFlow's ecosystem integration is valuable for deployment, the choice of normalization method should prioritize data characteristics over framework convenience, particularly in healthcare where outliers often carry clinical significance rather than representing noise to be suppressed.

4.16 Categorical Features & Feature Column API

This lesson demonstrates how to handle categorical features in Electronic Health Records (EHR) using TensorFlow's Feature Column API. The tutorial covers a three-step process for managing high-cardinality categorical data: first, creating vocabulary files containing unique values for categorical fields (with a placeholder for out-of-vocabulary values); second, using TensorFlow's categorical\_column\_with\_vocabulary\_file function to read from these vocabulary files; and third, converting the categorical features into one-hot encoded vectors using the indicator column function. The example uses a synthetic EHR dataset with principal diagnosis codes containing 6,752 unique values, demonstrating why vocabulary files are preferred over in-memory approaches for high-cardinality features. The lesson emphasizes that this pattern is particularly useful for features with large numbers of unique values and allows for decoupling vocabulary generation using tools like SQL. The approach transforms categorical data into binary vectors where only one element equals 1, corresponding to the specific category value, making it suitable for machine learning models in healthcare applications.

Vocabulary files don't inherently use less RAM than in-memory approaches. In fact, TensorFlow still needs to load the vocabulary file contents into memory to perform the categorical-to-numeric mapping during training.

The real advantages of vocabulary files for high-cardinality features are:

1. Maintenance & reproducibility: Versioned vocabulary files ensure consistent category-to-ID mappings across different runs and team members
2. Pipeline integration: External tools (SQL, Spark, etc.) can generate and update vocabularies independently of the model code, enabling better workflow separation
3. Flexibility for embeddings: Easy transition between one-hot encoding and embedding columns without modifying hard-coded lists in your code

The lesson's emphasis on vocabulary files for high-cardinality features isn't about RAM efficiency, but rather about operational benefits - maintainability, reproducibility, and integration flexibility in production ML pipelines. This is particularly valuable in healthcare applications where diagnosis codes and medical vocabularies frequently change and need to be managed consistently across different systems and teams.

4.18 Categorical Features with Feature Column API Solution

The video "Categorical Features with Feature Column API Solution" from Udacity’s AI for Healthcare nanodegree walks viewers through an exercise on feature engineering for Electronic Health Record (EHR) data, with a specific focus on handling categorical variables using TensorFlow’s Feature Column API. Unlike earlier lessons that employed indicator columns, this exercise requires the use of an embedding column, building a 10-dimensional embedding for the principal diagnosis code field. The instructor first reviews the provided code, which handles ETL tasks such as loading the dataset and constructing the vocabulary file for the categorical feature. The workflow involves creating a directory for vocabulary storage, generating the vocab file, and then defining a TensorFlow feature column to link the vocab file to an embedding feature. The key distinction in this solution is specifying the embedding’s dimensionality, resulting in each categorical value being represented by a 10-dimensional vector. This approach enables the model to learn dense, informative representations of categorical data, enhancing downstream predictive tasks. The exercise is intended as a practical review of previous material, with helper functions streamlining vocabulary file creation and embedding setup. Overall, the video clearly demonstrates the steps for implementing categorical feature embeddings in TensorFlow, reinforcing concepts essential for effective EHR data transformation.

A 10-dimensional embedding is used for the principal diagnosis code field instead of one-hot encoding to address the limitations of one-hot encoding when dealing with high-cardinality categorical variables. One-hot encoding creates a binary vector as long as the number of unique codes, which can result in very large, sparse vectors if there are many possible diagnosis codes—this is inefficient and can make learning relationships between codes difficult for machine learning models. In contrast, embeddings map each unique code to a dense, low-dimensional vector (in this case, 10 dimensions) that is learned during model training. These embeddings allow the model to capture complex relationships and similarities between different diagnosis codes by placing related codes closer together in the embedding space. This not only reduces memory and computational requirements but also often improves model performance for tasks like classification or prediction in EHR data. Thus, embeddings are preferred for categorical fields with many unique values, such as diagnosis codes, while one-hot encoding is more suitable for features with just a few categories.

4.19 Transformations and Feature Engineering Recap

In the "Transformations and Feature Engineering Recap" video from Udacity’s AI for Healthcare nanodegree, the instructor summarizes the key skills and concepts covered in the lesson on processing Electronic Health Record (EHR) data. The lesson emphasizes the importance of identifying the correct dataset level—line, encounter, and longitudinal—so that machine learning models are built using appropriate data granularity. Learners are guided through basic data splitting techniques that help prevent data leakage, a critical challenge in working with EHR datasets. The lesson then moves into hands-on feature engineering using TensorFlow, where students use the Dataset API for ETL (Extract, Transform, Load) operations and leverage the Feature Column API to efficiently process both numerical and categorical features at scale. This enables the transformation of raw EHR data into meaningful features suitable for machine learning. By the end of the lesson, students are equipped with foundational techniques in data transformation and feature engineering, setting the stage for the next phase: building and evaluating machine learning models for healthcare applications.

**Part 5: Building, Evaluating and Interpreting Models**

I am unable to directly edit Google Docs from here. However, you can easily copy and paste the following formatted summary into your "AI\_for\_HealthCare\_NanoDegree" Google Doc under Part 5, section 5.1:

### 5.1 Building, Evaluating & Interpreting Models Overview

This section marks the transition from preparing EHR data to the critical steps of building, evaluating, and interpreting machine learning models, with a strong emphasis on managing bias and uncertainty. The lesson begins by introducing hands-on experience with TensorFlow DenseFeatures, guiding learners through the construction of a simple regression model and demonstrating how to integrate feature column outputs. Next, it covers essential evaluation metrics specific to EHR models, including the implementation of Brier scores, which are particularly relevant for uncertainty estimation. The lesson then delves into demographic bias analysis, introducing the Aequitas framework from the University of Chicago for assessing group bias and fairness disparity. Following this, learners are exposed to uncertainty estimation using the TensorFlow Probability API, along with foundational concepts in Bayesian probability and types of uncertainty that influence model performance. The final segment focuses on model interpretability, first reviewing general approaches and then providing practical application with Shapley values, a robust model-agnostic interpretability method. Notably, some covered TensorFlow features and APIs are still in beta, reflecting the cutting edge of AI tools for healthcare applications. This overview establishes the foundation for advanced topics in evaluating and understanding models within the context of EHR data, ensuring responsible and effective AI deployment.

TensorFlow dense features are a type of input feature used in machine learning models where each feature is represented by a fixed-length, continuous numerical value. Unlike sparse features, which are typically used for categorical data with many possible values, dense features are ideal for representing structured numerical data such as age, height, or laboratory measurements. In TensorFlow, the tf.keras.layers.DenseFeatures layer enables seamless integration of these features into machine learning pipelines by automatically processing and combining them into a single dense tensor, ready for use in downstream neural network layers. This approach simplifies feature engineering and ensures efficient model training by handling data normalization and batching internally.

### 5.2 TensorFlow regression model with DenseFeatures

The video "Tensorflow Regression Model with DenseFeatures" from Udacity’s "Applying AI to EHR Data" nanodegree provides a concise overview of building a simple regression model using TensorFlow’s DenseFeatures layer. It explains that DenseFeatures is used to combine various feature columns—specifically numeric, embedding, bucketized, and indicator columns—into a dense representation suitable for input into neural network models. The lesson emphasizes that only these types of TensorFlow Feature Columns are compatible with DenseFeatures, and highlights the use of the Sequential API for model construction due to its simplicity, while also noting that the Functional API can be explored for more advanced customization, albeit with potential complications when integrating TensorFlow Probability outputs. The video briefly mentions ongoing experimental support for SequenceFeatures in TensorFlow, which is not covered in the course. To encourage further exploration, the lesson provides links to additional AutoML resources such as Google Cloud AutoML, AWS Autopilot, AdaNet, AutoKeras, and H2O Driverless AI. Overall, this section aims to equip learners with the foundational skills to build and evaluate regression models using structured EHR data, while providing pointers for more advanced modeling approaches.

5.3 TF DenseFeatures Walkthrough

The "TF DenseFeatures Walkthrough" video provides a practical demonstration of building a regression model using TensorFlow's DenseFeatures layer, with a focus on processing structured data for machine learning applications. Using the car mileage dataset from the TensorFlow regression tutorial, the walkthrough guides viewers through the steps of preprocessing the data, including normalization of numerical features and one-hot encoding of a categorical feature (country of origin). The video emphasizes the use of TensorFlow's feature column API to define and transform both categorical and numerical data, and shows how to combine these into a single dense input tensor using the DenseFeatures layer. This layer is then incorporated as the first layer in a Keras Sequential model, followed by simple fully connected layers with ReLU activations. The walkthrough highlights the use of common regression metrics such as Mean Squared Error (MSE) for model evaluation and notes that the provided model structure is not optimized but serves as an end-to-end example. Viewers are encouraged to experiment with the notebook to improve performance and further explore TensorFlow’s feature engineering and model-building tools for EHR and tabular data problems.

5.4 Common EHR Model Evaluation Metrics

The "Common EHR Model Evaluation Metrics" video reviews key metrics for assessing machine learning models in the context of Electronic Health Records (EHR) data. For classification models, it explains several foundational metrics: the AUC-ROC curve (Area Under the Receiver Operating Characteristic Curve) measures the ability of a model to distinguish between classes by plotting the true positive rate against the false positive rate at various thresholds; precision reflects the fraction of relevant instances among the retrieved instances; recall is the fraction of relevant instances that were actually retrieved; and the F1 score is the harmonic mean of precision and recall, providing a balanced assessment when there is an uneven class distribution. For regression models, the video highlights RMSE (Root Mean Square Error), which gauges the average magnitude of errors between predicted and actual values; MAE (Mean Absolute Error), which is the average of the absolute differences; and MAPE (Mean Absolute Percentage Error), which expresses errors as a percentage. The video also introduces the Brier Score, primarily used for evaluating probabilistic predictions in binary classification tasks. The Brier Score is defined as the mean squared difference between predicted probabilities and actual binary outcomes, where a lower score indicates better calibration and accuracy. The importance of the precision-recall tradeoff is emphasized, especially in healthcare, where balancing false positives and false negatives is crucial. Overall, the video provides essential definitions and practical context for selecting and interpreting evaluation metrics in healthcare AI applications.

5.5-5.6 Common EHR Model: Evaluation Metrics Solution

The "Common EHR Model Evaluation Metrics Solution" video guides learners through building and evaluating a regression model to predict resting blood pressure (trestbps) using EHR data. The exercise demonstrates merging two UCI heart disease datasets (Cleveland and Swiss), preparing the data by handling missing values, and selecting key features such as age and sex. The video emphasizes using TensorFlow’s DenseFeatures (note: might be outdated or deprecated) and introduces creating cross features by combining demographic variables (age and sex), including bucketizing age and one-hot encoding gender. The workflow involves converting data for modeling, splitting into train/test sets, and constructing bucketed and categorical features for use in TensorFlow’s feature column API, including cross features with hash buckets to manage computational complexity. The model is trained and evaluated, with predictions converted to a binary outcome using a threshold (e.g., blood pressure above 130). The video reviews evaluating the model using both regression and classification metrics and highlights the simplicity of the approach, encouraging further feature engineering for improved performance. Ultimately, the video provides an end-to-end walkthrough of model building, feature engineering, and evaluation, illustrating practical application of machine learning concepts to EHR data.

5.7-5.10 Demographic Bias Analysis

The "Demographic Bias Analysis" video from Udacity's "Applying AI to EHR Data" course emphasizes the critical importance of assessing and addressing demographic bias in healthcare machine learning models. It explains that while building and evaluating a model for accuracy and precision are essential steps, ensuring that the model is unbiased is equally crucial—particularly because biased models can restrict access to vital medical benefits or government aid, and may result in treatments that do not generalize well across diverse populations. The video introduces the concept of unintended or unconscious bias, which often goes unnoticed by model developers but can have significant impacts on certain groups. To combat this, the Aequitas framework, developed by the University of Chicago's Data Science for Social Good group, is highlighted as a tool for measuring and addressing group bias and fairness disparity. The video provides examples, such as diagnostic tests with higher false positive rates for specific races or eligibility models favoring certain age groups, to illustrate real-world consequences of demographic bias. Ultimately, the lesson stresses the need for awareness, transparency, and appropriate metrics in evaluating and mitigating demographic bias, thereby supporting fairness and equity in healthcare AI applications.

**Unintended biases:** a bias that is not intentional and often is not even apparent to the creator of a model. Unintended biases represent the unconscious or unintentional biases that come with the Al models that we are building. Becoming more aware of these biases and how they impact different groups is key.

Note: We usually associate bias with a negative connotation, but biases can be a source of valuable prior information. The problem can be when we are not aware of our biases and do not account for those that have a significant impact on the populations these models serve.

In the "Demographic Bias Analysis Walkthrough" video from Udacity's "Applying AI to EHR Data" course, viewers are guided through the essential steps of analyzing demographic bias in machine learning models using the Aequitas toolkit. The walkthrough focuses on a model predicting car MPG, treating the cars' country of origin as the demographic group. The process involves three key steps: first, selecting the demographic group for analysis (here, the 'origin' field); second, preparing the data using Pandas and Aequitas preprocessing functions, including handling missing values; and third, analyzing group-specific metrics. The video demonstrates visualization of the True Positive Rate (TPR), showing similar recall across the three countries of origin, indicating no apparent bias. It also examines the False Positive Rate (FPR), revealing a higher FPR for Japanese cars, which may signal specific characteristics influencing the model’s errors. This approach is presented as a structured, reproducible method to identify, flag, and address potential demographic biases in model predictions, emphasizing the importance of fairness and representation in AI systems—skills that are foundational for later analysis of patient demographics in healthcare applications.

In the "Demographic Bias Analysis Solution" video, the instructor guides viewers through performing a fairness disparity analysis using the Aequitas toolkit and the Compas dataset, which originates from a 2016 ProPublica investigation into criminal risk assessment algorithms. The exercise focuses on evaluating algorithmic bias by selecting “Asian females under 25” as the reference group and analyzing metrics such as false positive rates across different demographic groups, including race, gender, and age. The video demonstrates how to preprocess the data, summarize metrics, and visualize disparities using Aequitas, highlighting that the reference group is normalized to one and other groups are shown as multiples of this baseline. The analysis reveals significant disparities: for example, African Americans are over five times more likely, and Hispanic and Caucasian groups over twice as likely, to be falsely identified compared to the reference group. The instructor emphasizes the importance of such tools for uncovering and understanding bias in predictive models, advocating for careful consideration of reference group selection and the use of quantitative fairness metrics in model evaluation. This approach helps ensure the responsible deployment of AI, particularly in sensitive domains.

5.11 Uncertainty Estimation:

Uncertainty estimation is a critical aspect of evaluating machine learning models, especially in sensitive fields like healthcare. This lesson explains that while classification models provide class probabilities, these are not the same as true confidence in a prediction. Using relatable examples, such as weather forecasts, the video highlights the importance of quantifying uncertainty—knowing not just the prediction, but how certain the model is about it. The lesson introduces probabilistic programming, focusing on Bayesian Neural Networks implemented through TensorFlow Probability, which blend Bayesian inference with deep learning and allow for uncertainty quantification. Bayesian approaches are particularly valuable when datasets are small but domain expertise is high, as is common in healthcare. The video reviews key Bayesian concepts, such as prior and posterior distributions, showing how prior knowledge is updated with new evidence to yield improved predictions. Unlike frequentist models that offer binary outcomes, Bayesian models provide both a prediction and a measure of certainty, which can be crucial for decision-making in high-stakes scenarios like clinical trials. The lesson concludes that uncertainty estimation is essential for transparent, trustworthy model evaluation and deployment.

5.12 Model Interpretability with Shapley Values:

The video "Model Interpretability with Shapley Values" from the Udacity AI for Healthcare course discusses the critical importance of model interpretability, particularly in the context of electronic health record (EHR) data. Interpretability is essential for identifying biases, debugging models, and ensuring transparency—especially in healthcare, where decisions have significant impacts and are subject to regulatory and public scrutiny. While simpler linear models are easier to interpret, they often lack the expressive power needed for complex problems, leading practitioners to seek interpretability techniques for more advanced, "black box" models. The video introduces model-agnostic interpretability methods, focusing on LIME and Shapley values. Shapley values, rooted in game theory, measure the marginal contribution of each feature to a model's prediction by evaluating all possible feature combinations. The walkthrough demonstrates using the SHAP library to calculate these values efficiently, including summarizing data with K-means clustering to reduce computational costs. Visualizations illustrate how features influence predictions both individually and across samples, highlighting their importance and direction of effect. The lesson emphasizes that Shapley values provide clear, quantitative feature attributions, helping practitioners understand and trust complex models in sensitive domains like healthcare.

Part 6: Project: Patient Selection for Diabetes Drug Testing

**Project Overview**

EHR data is becoming a key source of real-world evidence (RWE) for the pharmaceutical industry and regulators to make [decisions on clinical trials](https://www.fda.gov/news-events/speeches-fda-officials/breaking-down-barriers-between-clinical-trials-and-clinical-care-incorporating-real-world-evidence). You are a data scientist for an exciting unicorn healthcare startup that has created a groundbreaking diabetes drug that is ready for clinical trial testing. It is a very unique and sensitive drug that requires administering the drug over at least 5-7 days of time in the hospital with frequent monitoring/testing and patient medication adherence training with a mobile application. You have been provided a patient dataset from a client partner and are tasked with building a predictive model that can identify which type of patients the company should focus their efforts testing this drug on. Target patients are people that are likely to be in the hospital for this duration of time and will not incur significant additional costs for administering this drug to the patient and monitoring.

In order to achieve your goal you must first build a regression model that can predict the estimated hospitalization time for a patient and also provide an uncertainty estimate range for that prediction so that you can rank the predictions based off of the uncertainty range.

**Expected Hospitalization Time Regression and Uncertainty Estimation Model**: Utilizing a synthetic dataset (denormalized, with line level augmentation) built off of the UCI Diabetes readmission dataset, students will build a regression model that predicts the expected days of hospitalization time and an uncertainty range estimation.

This project will demonstrate the importance of building the right data representation at the encounter level, with appropriate filtering and preprocessing/feature engineering of key medical code sets. This project will also require students to analyze and interpret their model for biases across key demographic groups. Lastly, students will utilize the TF probability library to provide uncertainty range estimates in the regression output predictions to prioritize and triage prediction uncertainty levels.

In the end you will be creating a demographic bias analysis to detect if your model has any bias which we know can be a huge issue in working with healthcare data.

Project Instructions:

**Context:** EHR data is becoming a key source of real-world evidence (RWE) for the pharmaceutical industry and regulators to make decisions on clinical trials(opens in a new tab). You are a data scientist for an exciting unicorn healthcare startup that has created a groundbreaking diabetes drug that is ready for clinical trial testing. It is a very unique and sensitive drug that requires administering the drug over at least 5-7 days of time in the hospital with frequent monitoring/testing and patient medication adherence training with a mobile application. You have been provided a patient dataset from a client partner and are tasked with building a predictive model that can identify which type of patients the company should focus their efforts testing this drug on. Target patients are people that are likely to be in the hospital for this duration of time and will not incur significant additional costs for administering this drug to the patient and monitoring.

In order to achieve your goal you must build a regression model that can predict the estimated hospitalization time for a patient and use this to select/filter patients for your study.

**Expected Hospitalization Time Regression Model:** Utilizing a synthetic dataset(denormalized at the line level augmentation) built off of the UCI Diabetes readmission dataset, students will build a regression model that predicts the expected days of hospitalization time and then convert this to a binary prediction of whether to include or exclude that patient from the clinical trial.

This project will demonstrate the importance of building the right data representation at the encounter level, with appropriate filtering and preprocessing/feature engineering of key medical code sets. This project will also require students to analyze and interpret their model for biases across key demographic groups.

**Dataset**

Due to healthcare PHI regulations (HIPAA, HITECH), there are limited number of publicly available datasets and some datasets require training and approval. So, for the purpose of this exercise, we are using a dataset from UC Irvine that has been modified for this course. Please note that it is limited in its representation of some key features such as diagnosis codes which are usually an unordered list in 835s/837s (the HL7 standard interchange formats used for claims and remits).

https://archive.ics.uci.edu/ml/datasets/Diabetes+130-US+hospitals+for+years+1999-2008(opens in a new tab)

Data Schema The dataset reference information can be https://github.com/udacity/nd320-c1-emr-data-starter/tree/master/project/data\_schema\_references(opens in a new tab).

There are two CSVs that provide more details on the fields and some of the mapped values.

**Project Submission**

When submitting this project, make sure to run all the cells before saving the notebook. Save the notebook file as "student\_project\_submission.ipynb" and save another copy as an HTML file by clicking "File" -> "Download as.."->"html". Include the "utils.py" and "student\_utils.py" files in your submission. The student\_utils.py should be where you put most of your code that you write and the summary and text explanations should be written inline in the notebook. Once you download these files, compress them into one zip file for submission in the Udacity Classroom.

Environment Setup

For step by step instructions on creating your environment, please go to https://github.com/udacity/cd0372-Applying-AI-to-EHR-Data(opens in a new tab)

Learning Objectives

By the end of the project, you will be able to:

* Use the Tensorflow Dataset API to scalably extract, transform, and load datasets and build datasets aggregated at the line, encounter, and patient data levels(longitudinal)
* Analyze EHR datasets to check for common issues (data leakage, statistical properties, missing values, high cardinality) by performing exploratory data analysis.
* Create categorical features from Key Industry Code Sets (ICD, CPT, NDC) and reduce dimensionality for high cardinality features by using embeddings
* Create derived features(bucketing, cross-features, embeddings) utilizing Tensorflow feature columns on both continuous and categorical input features
* Use the Tensorflow Probability library to train a model that provides uncertainty range predictions that allow for risk adjustment/prioritization and triaging of predictions
* Analyze and determine biases for a model for key demographic groups by evaluating performance metrics across groups by using the Aequitas framework

3. Steps to Completion

Please follow all of the directions in the Jupyter Notebook file in the classroom workspace or from the Github Repo if you decide to use your own environment to complete the project.

You complete the following steps there:

Data Analysis

* Create Categorical Features with TF Feature Columns
* Create Continuous/Numerical Features with TF Feature Columns
* Build Deep Learning Regression Model with Sequential API and TF Probability Layers
* Evaluating Potential Model Biases with Aequitas Toolkit

Project Submission

* When submitting this project, make sure to run all the cells before saving the notebook.
* Save the notebook file as "student\_project\_submission.ipynb" and save another copy as an HTML file by clicking "File" -> "Download as.."->"html".
* Include the "utils.py" and "student\_utils.py" files in your submission. The student\_utils.py should be where you put most of your code that you write and the summary and text explanations should be written inline in the notebook.
* Once you download these files, compress them into one zip file for submission in the Udacity Classroom.

Once you have completed your project please

* Make sure the project meets all of the specifications on the Project Rubric

If you are working in directly in our workspaces, you can submit your project directly there

If you are working in your own environment or if you have issues submitting directly in the workspace, please zip up your flies and submit them that way.

**Project Rubric:**

* **Exploratory Data Analysis**
  1. The project uses visualizations to analyze which fields have a high amount of missing/null values or high cardinality.
     + The project correctly identified which field(s) has/have a high amount of missing/zero values.
     + The project correctly identified which field(s) has/have a Gaussian distribution shape based on the histogram analysis.
     + The project correctly identified fields with high cardinality.
     + The project justified why these fields had high cardinality.
     + The project correctly describes the distributions for age and gender.
     + *Optional*: The project uses Tensorflow Data Validation Visualizations to analyze which fields have a high amount of missing/null values or high cardinality.
  2. The project selects features based on exploratory data analysis.
     + The project correctly identifies whether to include/exclude payer\_code and weight fields.
     + The project justified why these fields should be included/excluded by using supporting data analysis.
* **Data Preparation**
  1. The project uses the correct level(s) for the given EHR dataset (line, encounter, patient) and transforms, aggregates and filters appropriately.
  2. The project correctly reduces dimensionality for a dataset containing a high cardinality code set field.
     + The project correctly maps NDC codes to generic drug names and prints out the correct mappings in the notebook.
  3. The project dataset has been split correctly for EHR machine learning models.
     + The project has correctly split the original dataset into train, validation, and test datasets.
     + The projects dataset splits do not contain patient or encounter data leakage.
     + The project’s code passes the Encounter Test.
* **Feature Engineering**
  1. The project correctly creates categorical features using Tensorflow’s feature column API for transforming the data.
     + The project correctly completes the categorical feature transformer boilerplate function.
     + The project successfully uses this function to transform the demo dataset with at least one new categorical feature.
  2. The project correctly creates numerical features using Tensorflow’s feature column API for transforming the data.
     + The project correctly completes the numerical feature transformer boilerplate function.
     + The project successfully uses this function to transform the demo dataset with at least one new numerical feature.
     + The project's transformer function correctly incorporates the provided z-score normalizer function for normalization or another custom normalizer.
* **Model Building and Analysis**
  1. The project correctly utilizes the regression model predictions for uncertainty estimation and classification output analysis.
     + The project has prepared the regression model predictions for TF Probability and binary classification outputs by:
       - Correctly utilizing TF Probability to provide mean and standard deviation prediction outputs.
       - Creating an output prediction dataset that has the labels correctly mapped to a binary prediction and actual value.
  2. The project correctly evaluates the model predictions across key classification metrics.
     + The model has been evaluated across: AUC, F1, precision, and recall.
     + Students have completed both questions for the model summary and addressed bias-variance tradeoff in regard to this problem.
  3. The project utilizes the Aequitas toolkit to correctly create a bias report for race and gender on a provided dataset with justification for analysis.
     + The project contains a bias report with:
       - A visualization of at least two key metric(s) for patient selection.
       - A visualization showing at least one reference group fairness example and its comparison on at least one metric (e.g., TPR).
       - Justification for analysis made about at least one visualization.
* **Suggestions to Make Your Project Stand Out**
  1. Students can tune the TF Probability library to show and rank uncertainty estimates.
  2. Students can use cross-feature columns.
  3. Students can use ICD groupings to reduce dimensionality.
  4. Students can build a more complex representation of the data (e.g., conditional probability matrix, pre-trained embeddings, etc.).
  5. Students use Tensorflow Data Validation Visualizations to analyze which fields have a high amount of missing/null values or high cardinality.