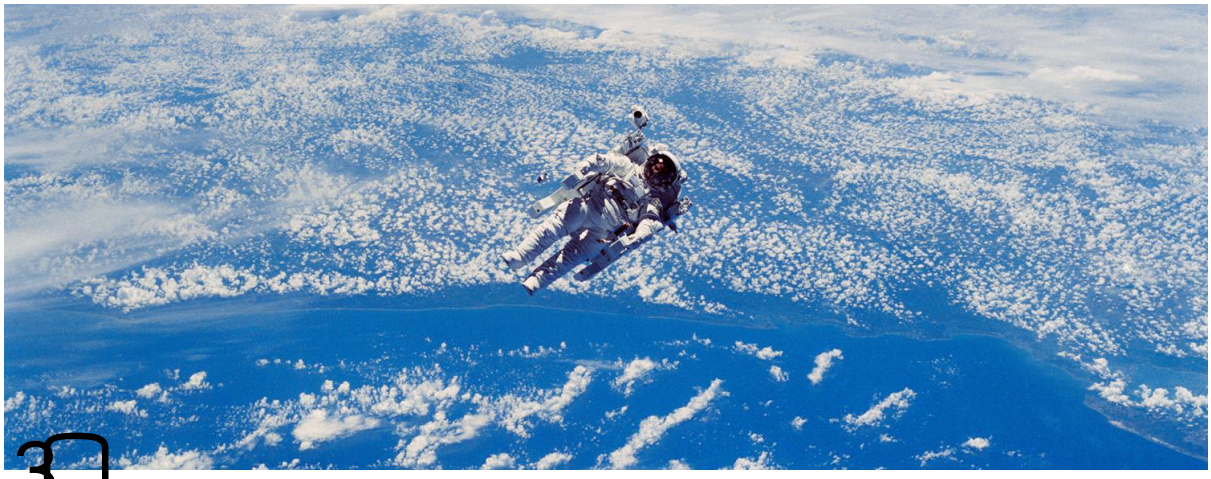


Appling AI to EHR Data

Applying AI to EHR Data Introduction

Historical Context of AI in EHR

Historical Perspective Key Points

Let's begin with what a healthcare record is since this is the basis for what is stored electronically is:

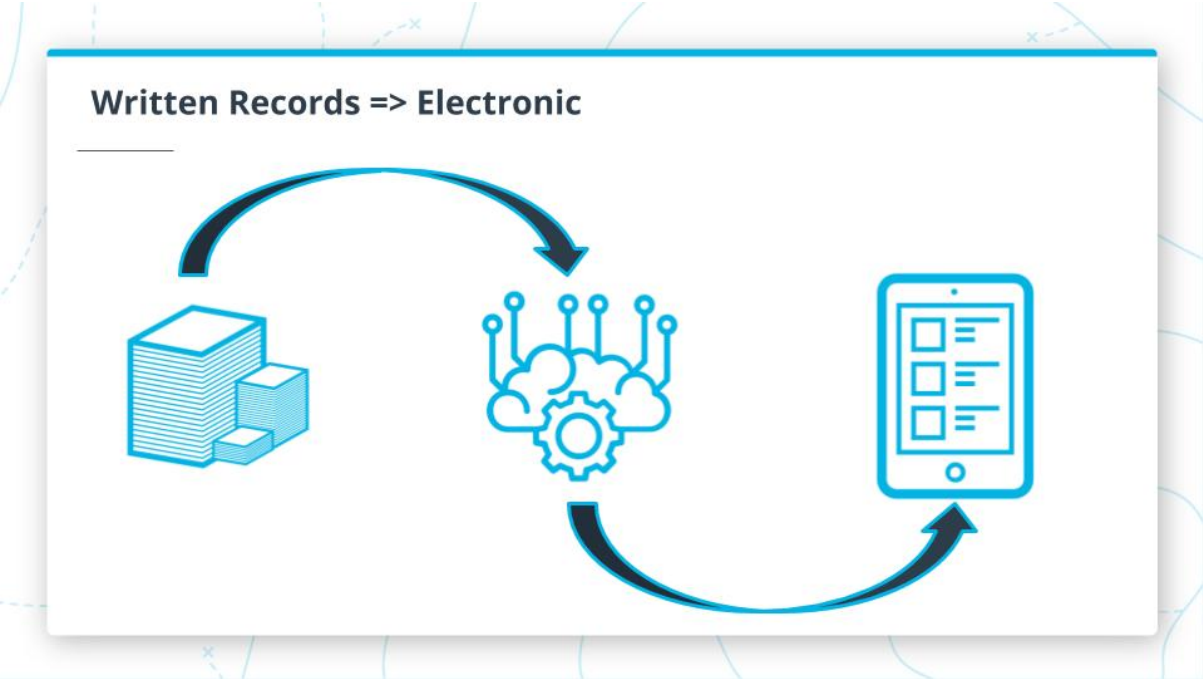
- **Health Record**: A patient's documentation of their healthcare encounters and the data created by the encounters across time
- **EHR**: Electronic Health Record

Health Records have been around for a very long time. As the need and capability arose to transition from paper to electronic medical records, EHR was born. There was even a law created to help this along. You might have heard of it. HIPAA, the Health Insurance Portability and Accountability Act, was passed in 1996 to address this. You'll learn more about this in the EHR Data Security section of the course.

- **HIPAA**: Health Insurance Portability and Accountability Act

EHR is also commonly referred to as

- **EMR**: Electronic Medical Record

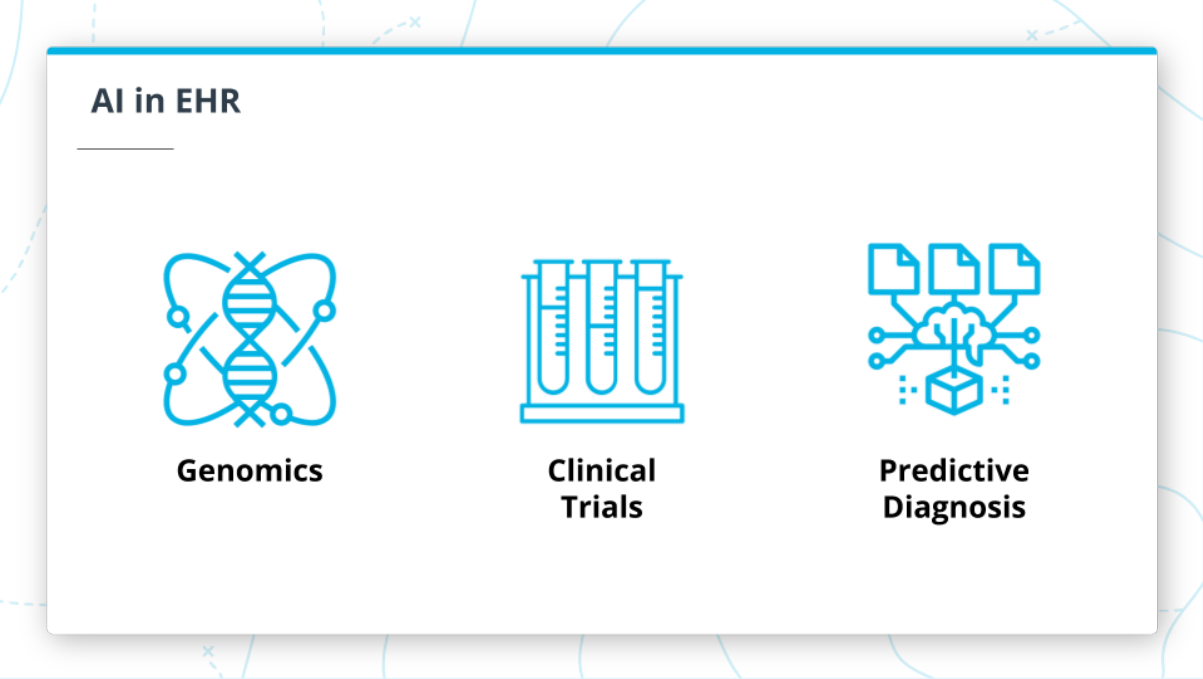


Written to Electronic Records

Moving from Written Records to Electronic Records

The transition from paper to electronic records not only led to brand new systems for storing and collecting EHR data but also for the need to translate all of the old paper data into something usable by these new systems.

That's a particularly great job for AI! But there are many many other potential uses for AI with EHR.



AI in EHR Opportunities

AI in EHR Opportunities

A few other fantastic uses of AI in EHR Include:

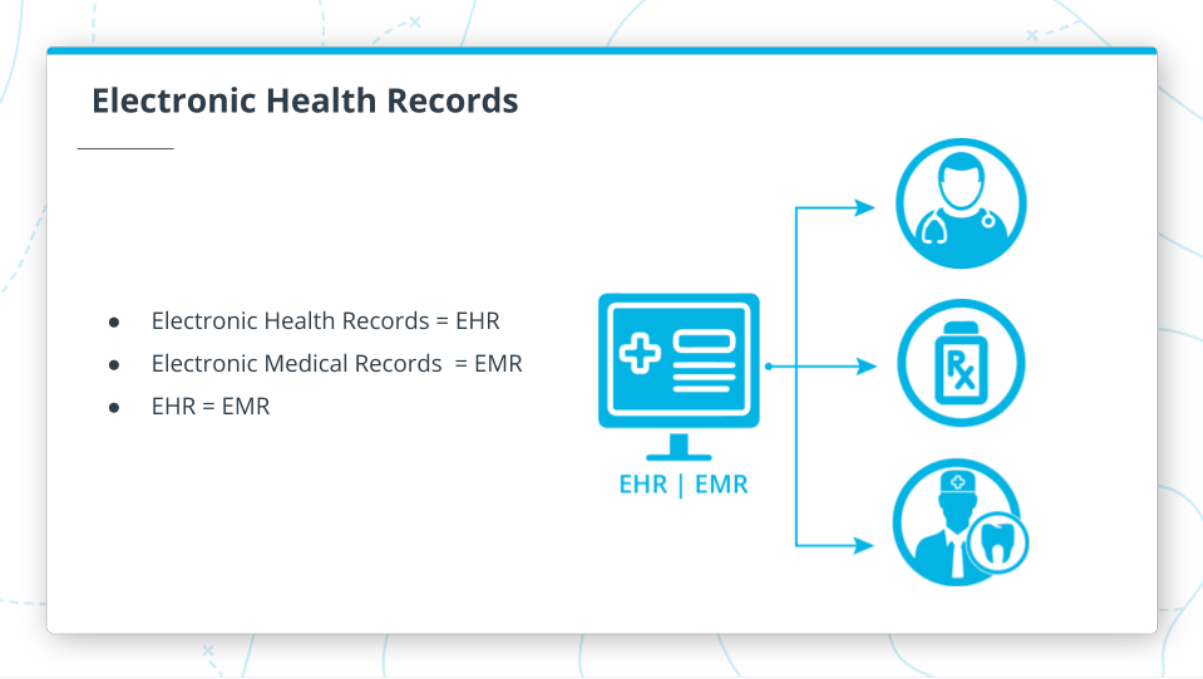
- Mapping of our genes in Genomics
- Analyzing data from clinical trials
- Predicting a diagnosis for patients

There are many others, but these are few that are seeing some of the most traction right now.

That's it for your history lesson on EHR. Now that you have a little background, we'll discuss the current landscape for EHR Data and AI.

Landscape of EHR Data

Importance of EHR Data Key Points



EHR: Electronic Medical Record or EMR: Electronic Medical Record

What does EHR stand for?

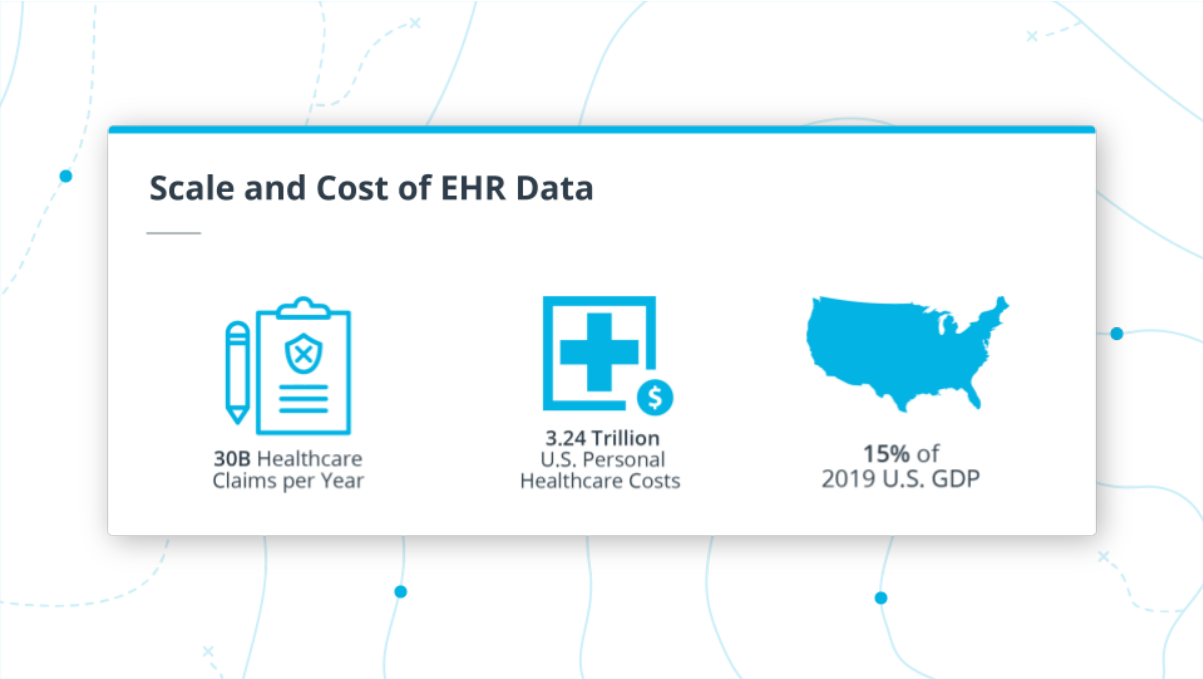
Again as a quick review from the last section, EHR stands for Electronic Health Records, which is synonymous with Electronic Medical Records, or EMR.

What is EHR data?

EHR Data is the data being collected when we see a doctor, pick up a prescription at the pharmacy, or even from a visit to the dentist. These are just a few of the examples where EHR data is collected.

This data is used for a variety of use-cases. From personalizing healthcare to discovering novel drugs and treatments to helping providers diagnose patients better and reduce medical

errors.



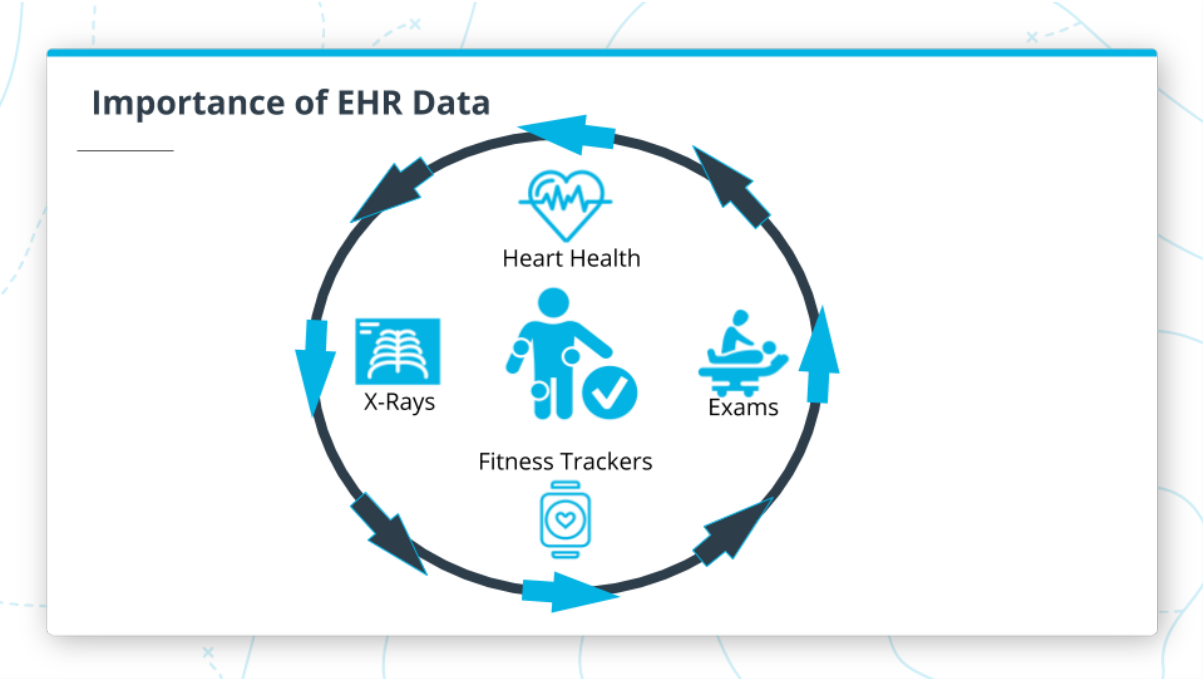
Scale and Cost of EHR Data

To illustrate the scale of EHR data in the U.S. there are some estimates of about 30B healthcare claims just last year! Regardless of the exact amount, anything in this range shows the large volume of claims that are being processed. In addition and related to the access is the high cost of healthcare in the U.S. for 2019, Personal Healthcare Costs in the U.S. was projected to be $3.24T by CMS. This is an enormous amount and to put that in perspective that 3T number is almost 15% of the U.S. GDP for 2019. You can learn more about all of this by looking through the links below.

Additional Resources

- Health Affairs
- CMS
- 50 Surprising Statistics Every Healthcare Stakeholder Must Know

Importance of EHR Data



Importance of EHR Data

EHR Data can come from many different sources in today's world. Data such as:

- Heart rate monitors
- Xrays and other radiology scans
- Fitness trackers
- Other diagnostic tests

All this helps to provide a 360-degree view of a patient's health. Ensuring that this data is collected in an electronic health record is very valuable to the patient and their caregivers. This data becomes even more critical when you can get a longitudinal view of a patient's data and collect it in a way that allows healthcare professionals and data scientists to make meaningful and accurate predictions.

To make this data even more impactful and create better insights and predictions, we need to be able to aggregate the data of many to help address some of the most significant opportunities in healthcare.

Many companies are doing great things with healthcare data including Google and Apple.

- Apple Healthcare Data
- Google Cloud Healthcare API
- Google Health

Opportunities

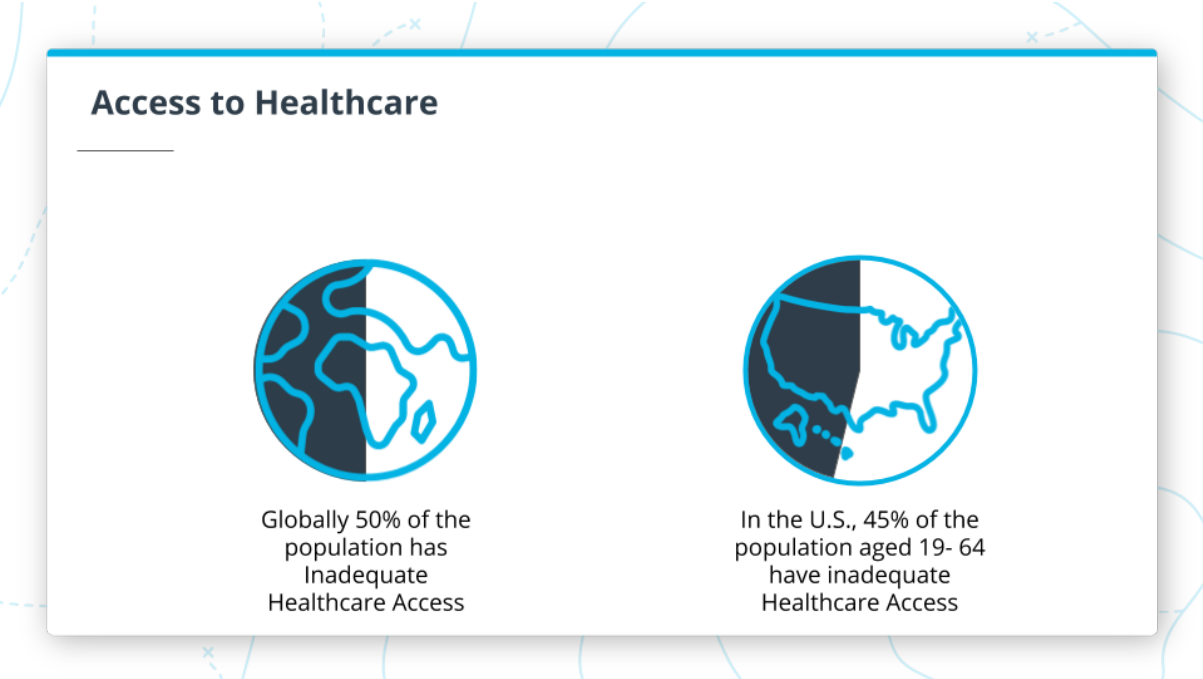
Opportunities

As mentioned earlier, there is a major opportunity for impact and with the shift to electronic health records over the last decade. This has enabled totally new ways to analyze and use EHR data. In particular, EHR data and AI will enable increased healthcare access by lowering healthcare costs. One of the major challenges in the U.S. healthcare system is reducing administrative errors and with AI and streamlined technology, we might eliminate many manual error-prone processes.

EHR data will also contribute to better diagnostics, companies are leveraging the large datasets on populations to find ways to predict diseases and conditions well before they occur by seeing trends that only these large datasets can provide.

Lastly, EHR will unlock more powerful treatments by combining EHR data with other data sources like genetics and imaging data to build even more powerful and personalized treatments. Companies are already using EHR data to identify patient cohorts and clinical trials more effectively and predicting risks for different drugs based on population data instead of only human expertise.

Access to Healthcare



Access to Healthcare

Despite all of the advances in technology for healthcare, there are many challenges still to overcome.

Globally, a couple of years ago, the WHO and World Bank released a report that stated that "half the world lacks access to essential health services." This issue of access can also be seen in a report by the NYU Commonwealth Fund that found that in the U.S., 45% of U.S. adults between 19 - 64 are inadequately insured. While these statistics might be subject to multiple interpretations, there is a collective agreement that access to quality healthcare is an issue for many globally and also in the U.S.

These are only **some** of the many opportunities of using AI with EHR data in healthcare. Hopefully, after completing this course, you will be able to apply your new skills to make a huge breakthrough in the medical world. I'm excited to see what new things happen in the field.

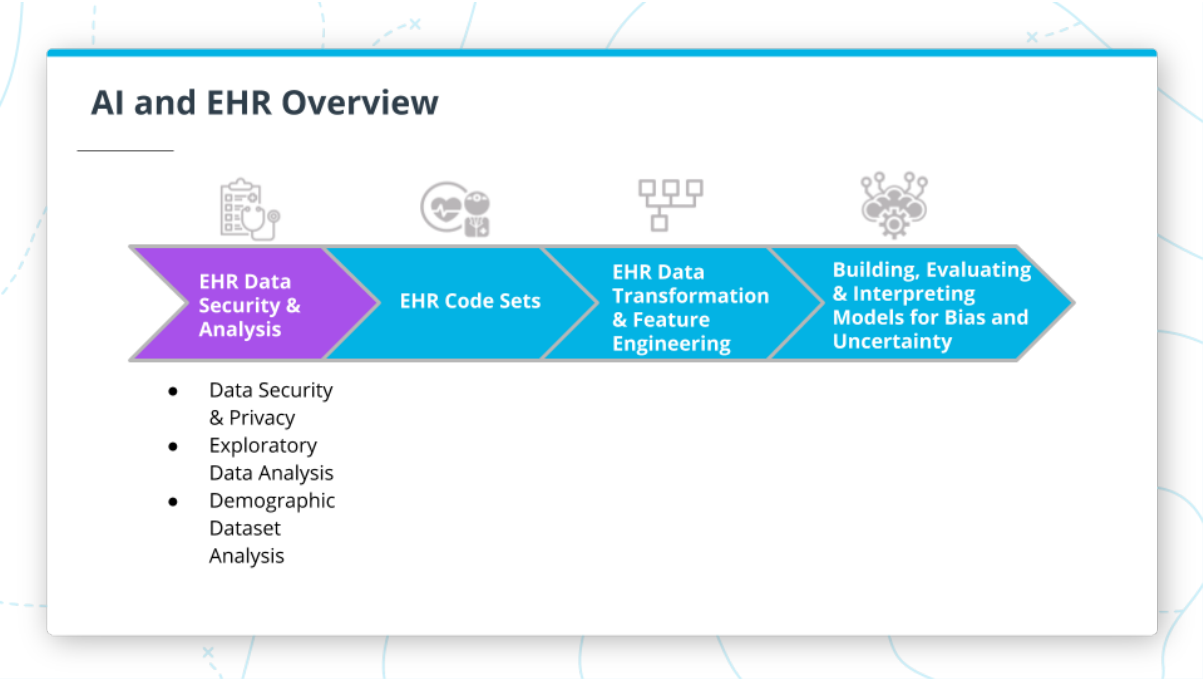
Additional Resources

- WHO and World Bank
- NYU Commonwealth Fund

AI and EHR Course Overview

Now that you have a little background into AI and EHR data, let's get a preview of what we will cover in this course.

EHR Data Security & Analysis



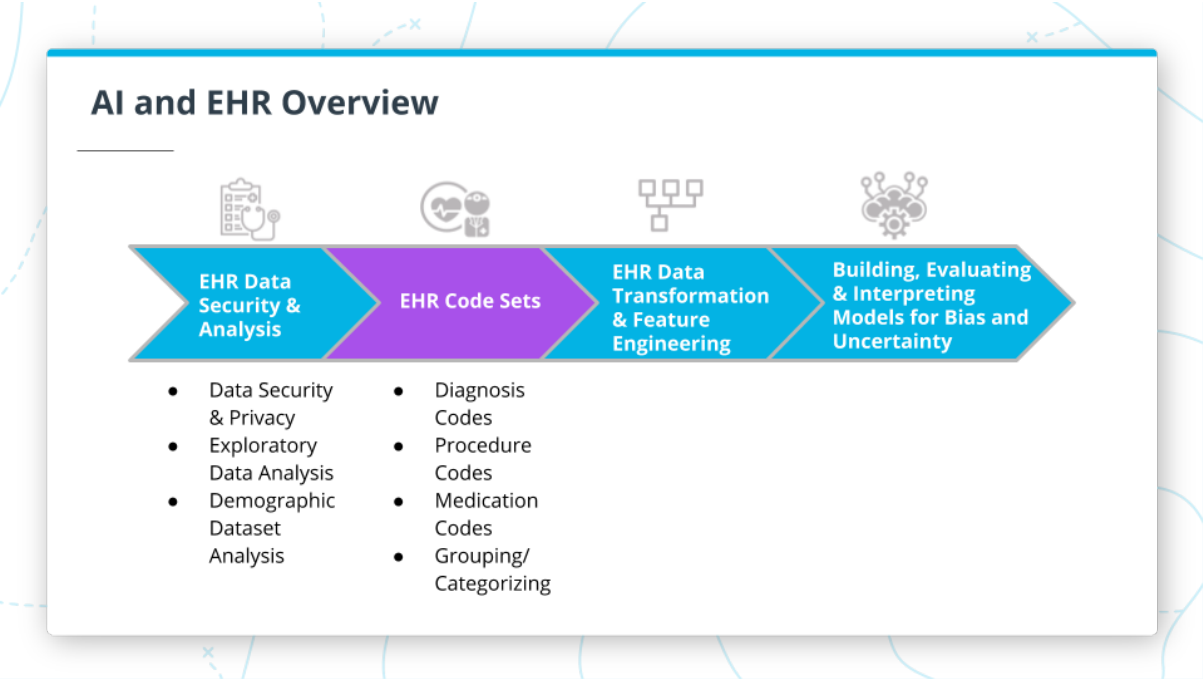
EHR Data Security & Analysis

In the initial parts of this course, we will be reviewing EHR Data security and Analysis.

This Data Security section of the course may be a review for anyone who currently works in healthcare, but it is of vital importance in today's world to understand how to keep our patients' data safe and secure. We are the gatekeepers of that data and if we want to be able to do amazing things with this data, we need to ensure it's safe. Of course, these rules also protect medical professionals as well.

You will learn about the different regulations around the world which you should be aware of to ensure your data practices meet these regulations. You'll also learn who generally takes care of making sure your data is compliant before it even gets to you.

We will be reviewing and completing some exploratory data analysis following the CRISP-DM framework to address some common issues with data before we begin using it to build and test our models. You will perform some data schema and demographic analysis as part of the section. This will enable you to explore your data and make sure it is representative of the population and that will allow the model you build later to be accurate.



EHR Code Sets

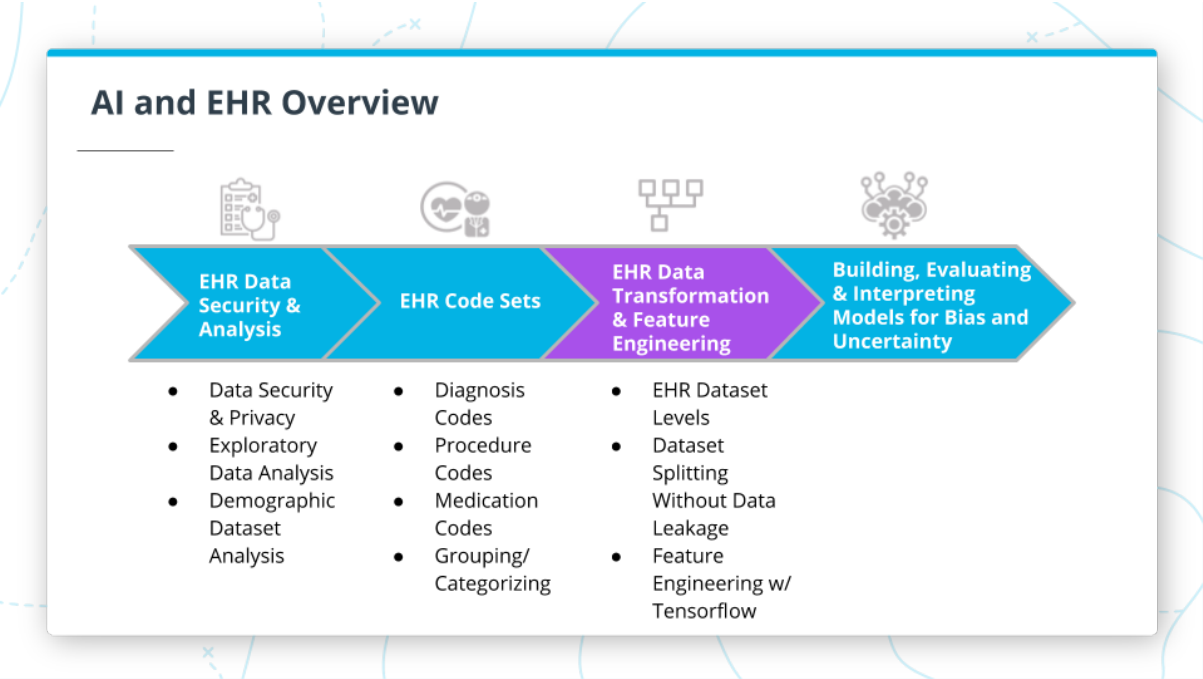
EHR Code Sets

In the EHR Code Sets section of this course, you will learn about all kinds of different code sets used in the medical industry. You'll become familiar with medical Encounters and learn to interpret Diagnosis, Procedure, and Medication codes.

Diagnosis codes are part of the ICD10-CM classification of diseases.

Then you'll learn to break down procedure codes using CD10 PCS codes and CPT categories.

Finally, we will look into using the Clinical Classifications Software or (CCS) system to more easily group and categorize these 77K+ codes when using them for our ML models.

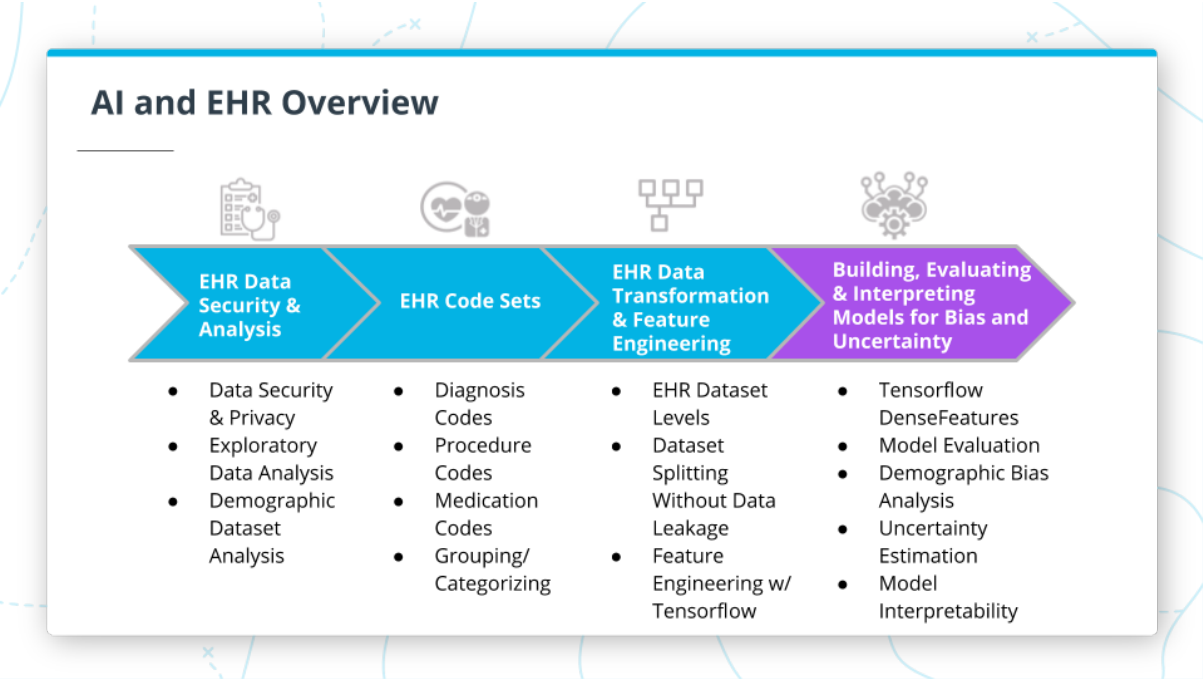


EHR Data Transformation and Feature Engineering

EHR Data Transformation and Feature Engineering

First, you'll learn to differentiate between and choose the correct Dataset level for your modeling by choosing between line, Encounter and Longitudinal levels.

Then you'll learn to apply testing and validation techniques to prevent data leakage and population misrepresentation during dataset splitting, which, if left unchecked, can lead to highly inaccurate models in production. And finally, you perform feature engineering use the TensorFlow Dataset API. Then it's time to get to the really fun part, Building, Evaluating and interpreting models!



Building, Evaluating & Interpreting Models

Building, Evaluating & Interpreting Models

In the last part of the course, it's time to get to the really fun part, Building, Evaluating and interpreting models!

You'll start off by getting a quick review of TensorFlow DenseFeatures and implementing best practices with TF Feature columns. Next, you'll evaluate your models for Uncertainty and Bias while being introduced to Brier Scores for evaluation.

After that, we'll head into a deep dive on demographic bais analysis of our models to remove bias. Not only will you understand why unintended demographic bias is so important to address, but also how to address unintended biases using a project called Aequitas.

Then, you'll get a review of uncertainty estimation and Bayesian probability before Training an Uncertainty Estimation Model with TF Probability.

Finally, we'll use Shapely values to help interpret your model.

Wow, looking at this course, it might seem like a lot, but by the time you have finished, you will have gains some amazing skills you can put to use in the ever-growing and important industry of healthcare!

Relevant Tools for AI and EHR

In this course, you will be using Jupyter Notebooks throughout. We will be working with some fantastic libraries like:

- Python
- Numpy
- Pandas

- Seaborn
- Matplotlib
- TensorFlow
- TensorFlow Probability
- Aequitas
- Shapley

To be able to complete this course, you need to have:

- Intermediate Python
- Basic understanding of machine learning

This is not an introductory course and you should have a solid foundation in these skills before taking this course.

Luckily, Udacity has many fantastic courses in these areas to get you up to speed if you need them!

While this course focuses on Python and Tensorflow there are many other tools you might be exposed to working with AI and EHR.

First, for really big data sets and more computing power, you might use one of the cloud providers like

- Google Cloud Platform
- Amazon AWS
- Azure

Some other popular languages for working with AI in healthcare might include:

- R
- Julia
- SQL (on cloud query engines such as GCP Big Query and AWS Athena)

Finally, other popular Machine Learning packages include:

- PyTorch
- Scikit-learn
- PySpark

While some of the features shown in this course are specific to TensorFlow, many of the important techniques you will learn will apply no matter where or how you create your models.

Finally, you may encounter some EHR data systems. Below are only a few of them.

- Cerner
- Epic

However, it is unlikely you would be working directly with these systems, but it's good to know about 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

EHR Data Security And Analysis

EHR Data Security and Analysis Lesson Overview

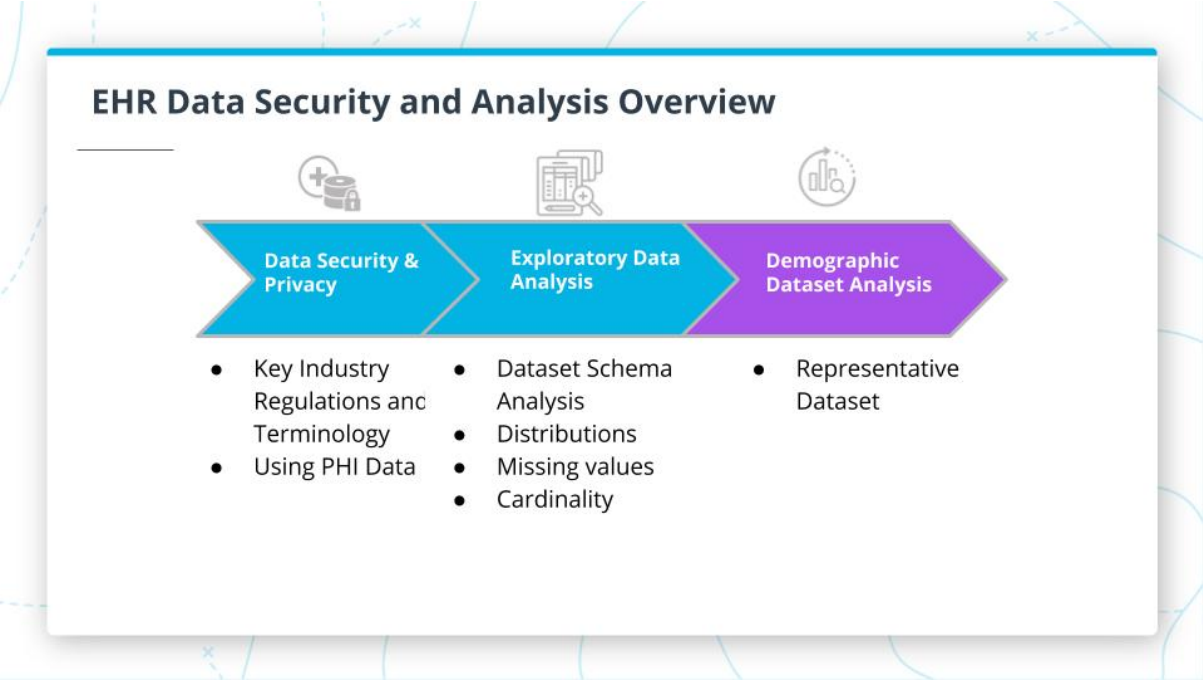
In this lesson, you will cover

- Data Security and Privacy, including some of the key standards and regulations.
- Exploratory data analysis allowing you to gain a deeper understanding of your datasets, including:
    - Dataset schemas

- Value distributions
- Missing values
- Cardinality of categorical features

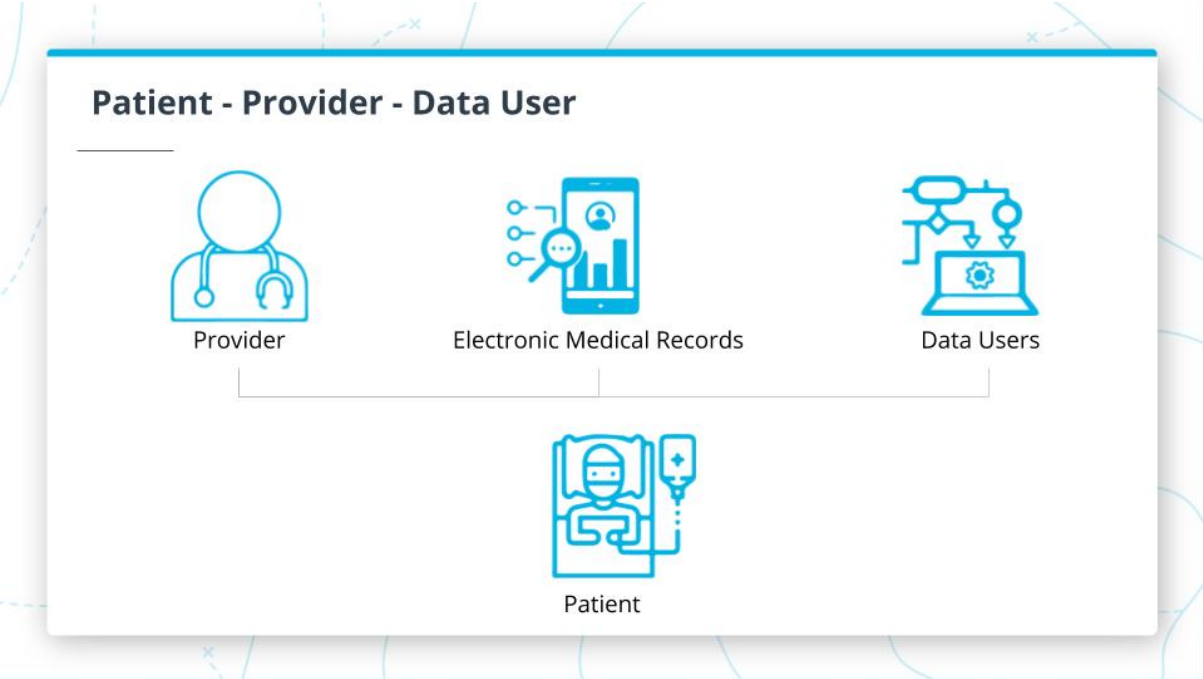
- Demographic dataset analysis to show how representative your dataset might be

With that let's get started on the journey towards AI and Electronic Healthcare Data (**EHR**)



EMR Data and Security Lesson Overview

Importance of Data Privacy and Security



Patient - Provider - Data User

Patients, Providers & Data Users

Patients are pretty obvious. They are the people seeking care.

What is a Provider?

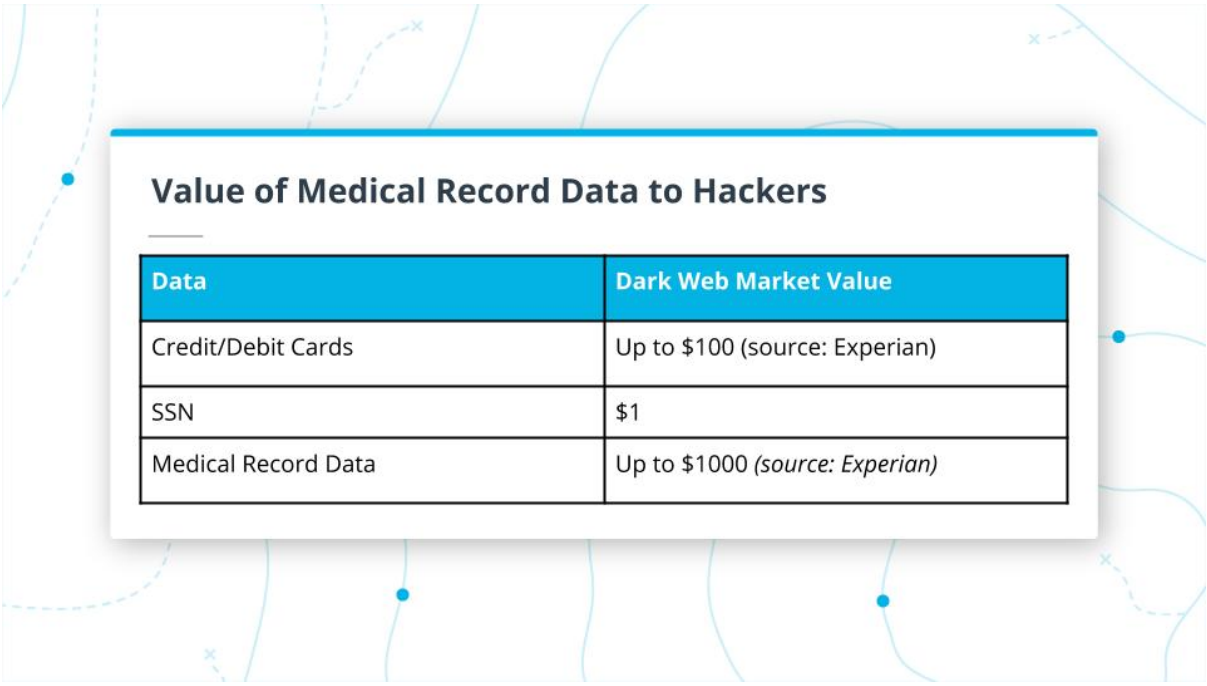
Providers : These are entities/groups/individuals that provide care for patients. Providers can range in size from entire hospital networks to individual doctors and medical professionals. Some examples are

- healthcare professionals treating you at a hospital
- in a doctor's clinic
- dentist office

This interaction between a patient and a provider results in the production of an **Electronic Health Record** or **EHR**. Electronic health records can be used by data users, too.

Who are some data users?

Payers: a group that consists of companies like healthcare insurance. Payers can also include entities such as the government for things like Medicaid and Medicare in the United States.



Medical Record Data is up to 1000 times more valuable to hackers than other data

Data Security From the Patient Perspective

Data security is crucially important for many different reasons in healthcare, but here are a couple of very important reasons:

1. Patients have a right to privacy and may not want to share their conditions and treatments with others.
2. Medical data is far more valuable to hackers. Because of the detail and complexity of data in EHR, hackers can very easily impersonate a patient. You can learn more about this in the links below.

Additional Resources

- Value of Medical Data on the Dark web
- Hacker Hone their Techniques

Data Security Importance Provider Perspective

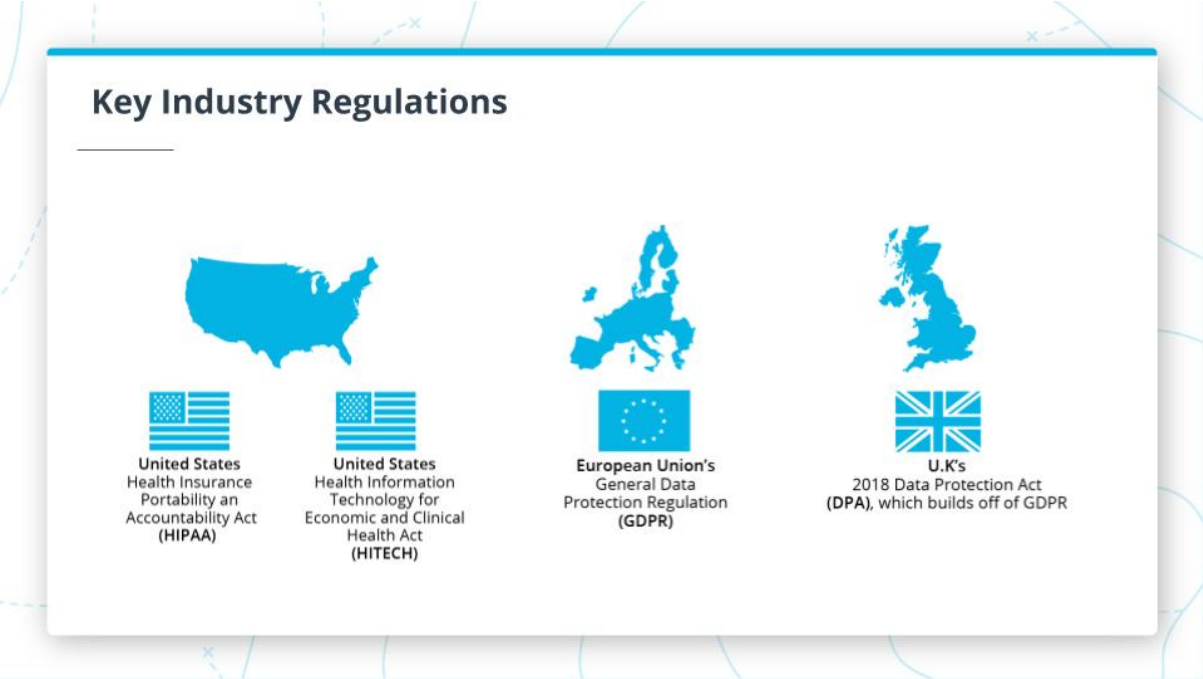
Medical professionals/organizations that are “providing” care to patients are under strict regulatory compliance laws. Your organization will handle the intricacies of how to interface with providers or if your organization is one, they have extensive training as part of the required HIPAA compliance. You'll learn more about HIPAA in a bit. To make sure organizations comply, there are some penalties and they can change periodically over time. Please check the HIPAA website to be up to date.

Examples from U.S. HIPAA fines

Importance from a Data User Perspective

Data Privacy issues come up with disclosure from hackers but also from legitimate entities within a business or medical research purposes. For example, the issue of informed consent is an essential part of any scientific experiment. Remember our examples of Deep Mind in the UK and Ascension in the U.S.

Key Healthcare Data Security and Privacy Standards



Key Industry Regulations

Key Industry Regulations

Healthcare data security and privacy regulations are ubiquitous around the world. Most countries have regulations regarding data security and privacy around electronic health records with varying levels of complexity. However, for this course, we will focus on U.S. healthcare regulations.

United States:

- **HIPAA**: The Health Insurance Portability and Accountability Act is the key industry regulation that you should be familiar within the U.S.
- **HITECH**: The Health Information Technology for Economic and Clinical Health Act is also important to note and this is really just an update to HIPAA that accounts for technology.

HIPAA was passed in 1996 and then updated in 2009 under the HITECH act to evolve for digital technologies. While it might seem like EHR has been around for a while, it was only a little over a decade ago when the foundational legislation really was put in place that made the healthcare community start moving towards digitizing and putting their records in electronic format. Even now many systems are legacy systems that are built off of transcribed

handwritten notes and other unstructured formats that organizations are spending large amounts of time to normalize and aggregate this info.

EU: European Union

- **GDPR**: The General Data Protection Regulation is generally considered more stringent than even HIPAA when it comes to protections for patients.

United Kingdom:

- **DPA**: The Data Protection Act really builds off of and add to GDPR

Additional Resources:

- HIPAA
- HITECH
- GDPR
- DPA

PHI

PHI: Protected Health Information

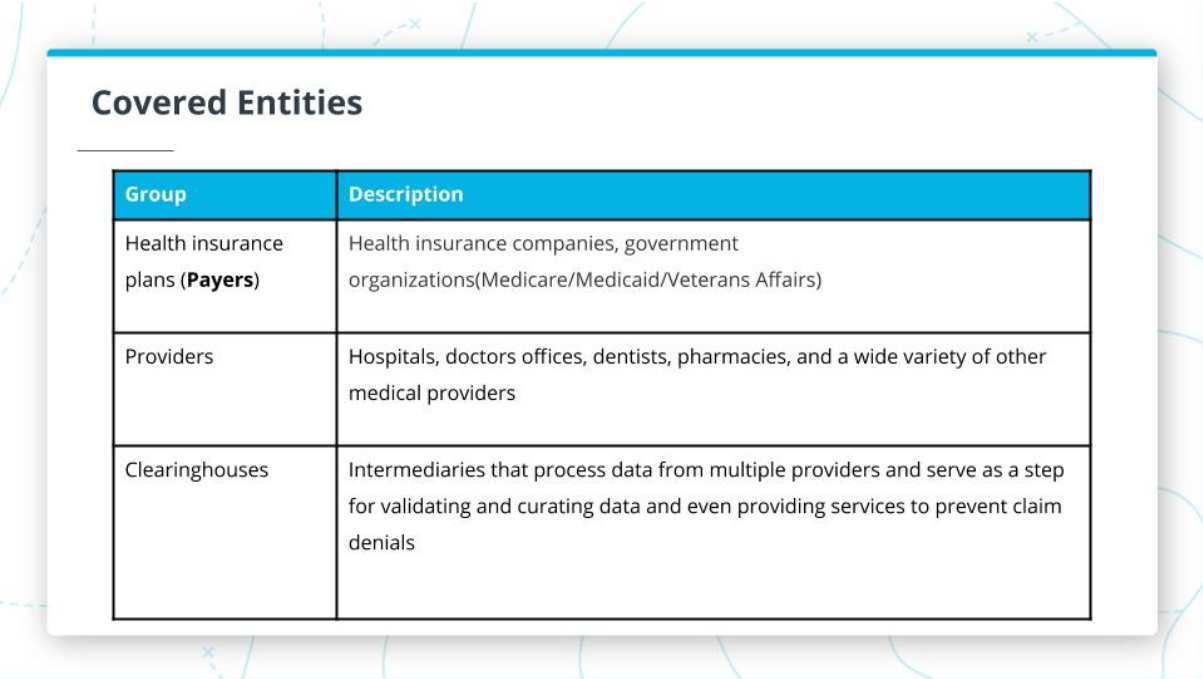
Probably one of the most important terms that you should be familiar with is PHI, which stands for protected health information. PHI is a part of HIPAA that protects the transmission of certain types of personally identifiable information such as name, address, and other info. Certain information in an electronic medical record is considered PHI and must comply with HIPAA standards around data security and privacy. This informs not only how you transmit and store data but also data usage rights and restrictions around building models for other purposes than the original use.

PHI by nature is identifiable information and can be used to easily identify a person. However, please note that **omitting this information does not ensure that a person can not be identified**.

Additional Resources

PHI

Covered Entities



Covered Entities

Group	Description
Health insurance plans (**Payers**)	Health insurance companies, government organizations(Medicare/Medicaid/Veterans Affairs)
Providers	Hospitals, doctors offices, dentists, pharmacies, and a wide variety of other medical providers
Clearinghouses	Intermediaries that process data from multiple providers and serve as a step for validating and curating data and even providing services to prevent claim denials

Covered Entities

Covered Entities: are a group of industry organizations defined by HIPAA to be one of three groups: health insurance plans, providers, or clearinghouses. You can see from the table the types of entities in each category.

These groups transmit protected health information and are subject to HIPAA regulations regarding these transmissions.

Other Covered Entities:

- **Business Associates**: A business associate is a person or entity that performs certain functions or activities that involve the use or disclosure of protected health information on behalf of, or provides services to, a covered entity.
  - Covered entities can disclose to BAs under Privacy Rule
    - Only for a purpose allowed by the covered entity
    - Safeguard data against misuse
    - Comply with other requirements of the covered entity under HIPAA Privacy Rule
- **Business Associates Agreement/Addendum (BAA)**: this is the contract between a covered entity and BA

Storing and Accessing PHI Data

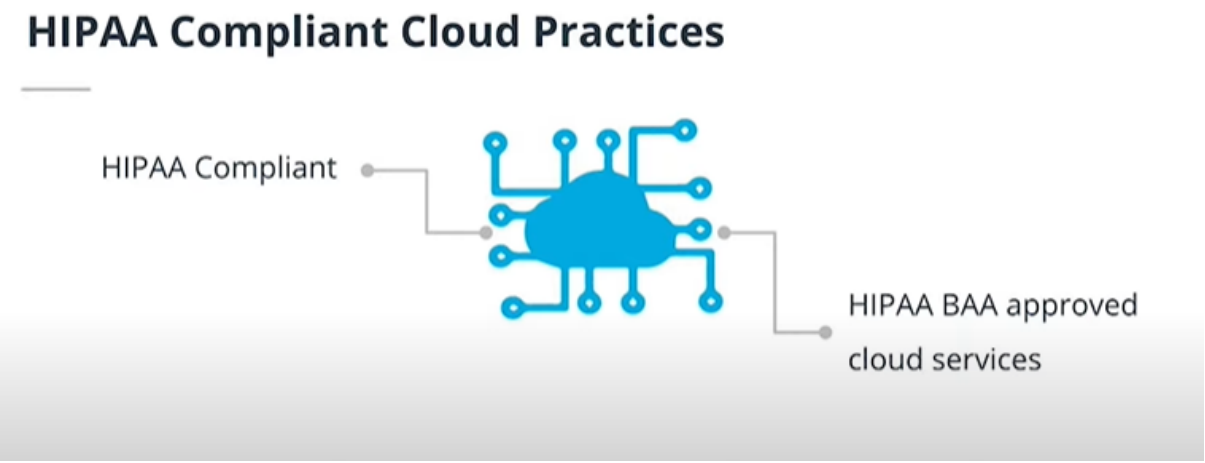
Securing PHI Data

- **Not Required** under HIPAA security Rule
  - **Considered Best Practice**
- Data at rest

- VPN and secure transmission of data
- Theft/loss of physical devices with PHI such as laptops, mobile devices

Minimum Necessary Standard

- Covered entities must make reasonable efforts to limit PHI access to the minimum necessary level to accomplish objective.
- Greater Oversight on use
- Limited latitude on how to use the data



Minimum Necessary Standard Examples

- Add additional fields
- Use identifiable data to add a filter to records

De-identifying a Dataset

De-identifying a dataset refers to the removal of identifying fields like name, address from a dataset. De-Identification is done to reduce privacy risks to individuals and support the secondary use of data for research and such.

This is not something you should be doing on your own. HIPAA has two ways that you can use to de-identify a dataset.

The first method is the **Expert Determination Method** and this is done by a statistician that determines there is a small risk that an individual could be identified.

The second method is called **Safe Harbor** and it refers to the removal of 18 identifiers like name, zip code, etc.

Limited Latitude: Very limited scope of work. EHR Data can only be used for the purpose granted.

Additional Resources

De-Identification Rationale

PII Data Solution

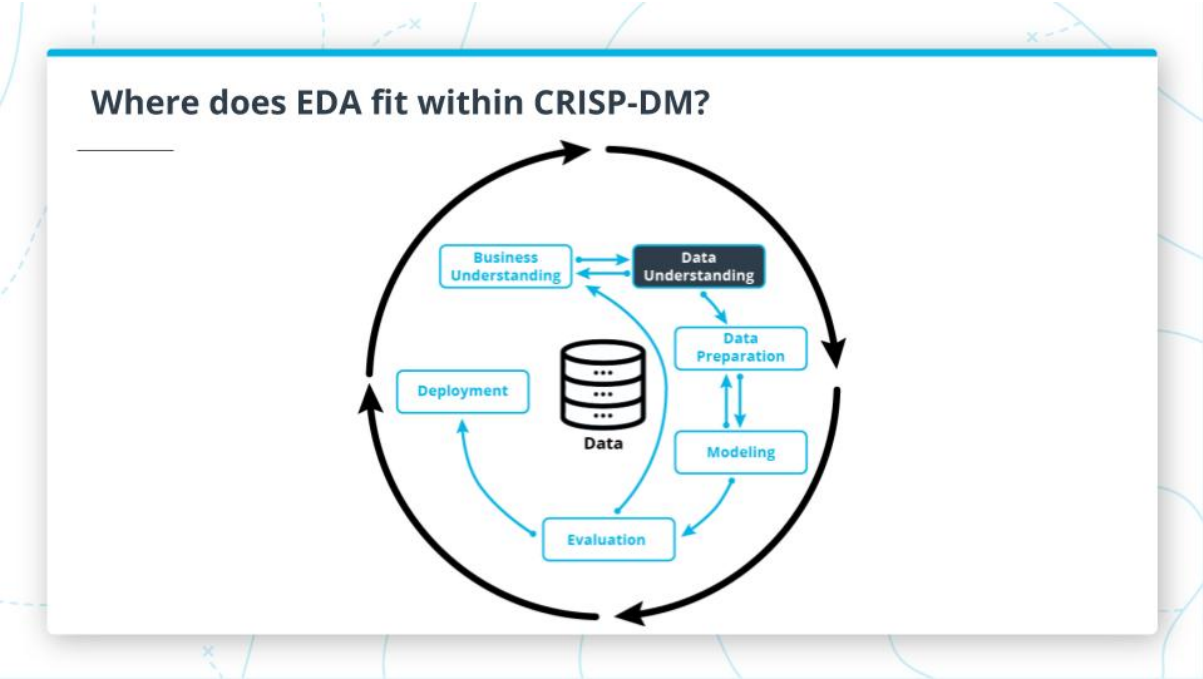
Key Takeaways

- Be aware of what you do not know
- Your organization likely has compliance protocols and rules
- HIPAA training for those in the U.S.

Personal Interest

- The way privacy is evolving
- How methods are gaining more acceptance across different fields
- How this will evolve in healthcare

Importance of EDA



Where EDA fits into CRISP-DM

Data Schema Analysis

EDA: Exploratory Data Analysis

EDA is a step in the data science process that is often overlooked for the modeling and evaluation phase that can be easier to quantify and benchmark.

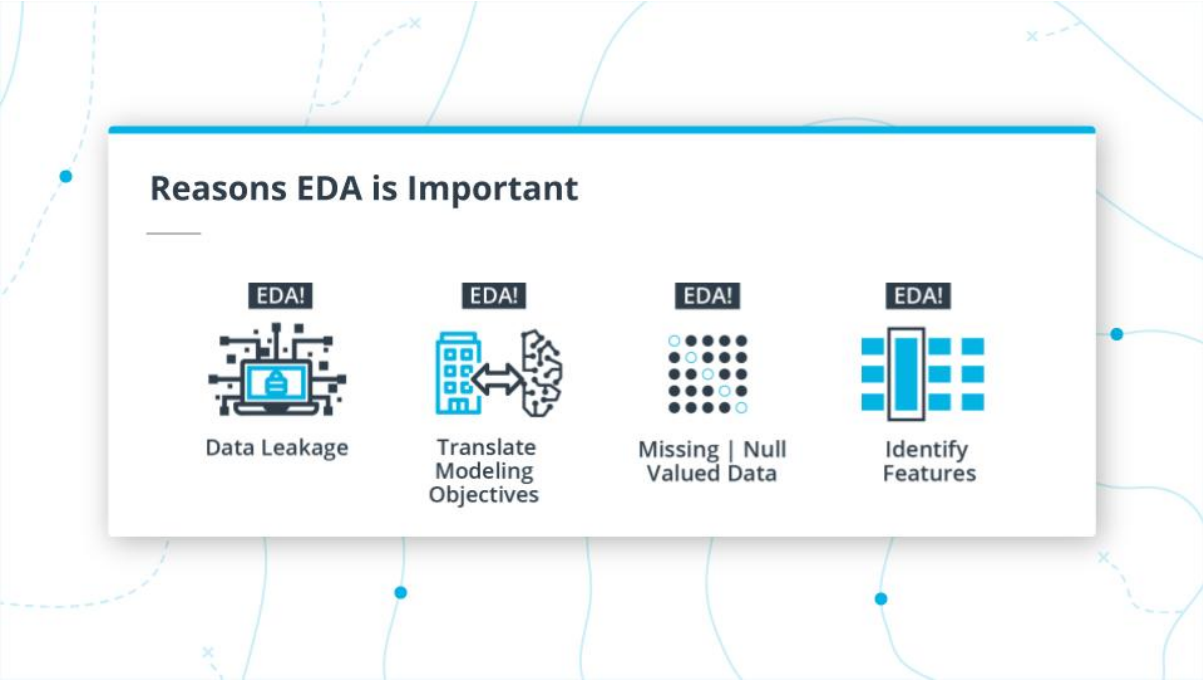
CRISP-DM: This stands for “cross-industry standard process for data mining” and is a common framework used for data science projects and includes a series of steps from business understanding to deployment.

EDA and CRISP-DM

As you can see from the image above EDA falls in the Data Understanding phase of CRISP-DM

Additional Resources

CRISP-DM What is Exploratory Data Analysis EDA in Python



Reasons EDA is important

- EDA can enable you to discover features or data transformations/aggregations that might have data leakage. This can save a tremendous amount of time and prevent you from building a flawed model.
- EDA can help you better translate and define modeling objectives and corresponding evaluation metrics from a machine learning/data science and business perspective.
- EDA can help inform strategies for handling missing/null/zero valued data. This is a common issue that you will encounter with EHR data that you will have missing values and have to determine imputing strategies accordingly.
- EDA can help to identify subsets of features to utilize for feature engineering and modeling along with appropriate feature transformations based off of type (e.g. categorical vs numerical features)

Dataset Schema Analysis

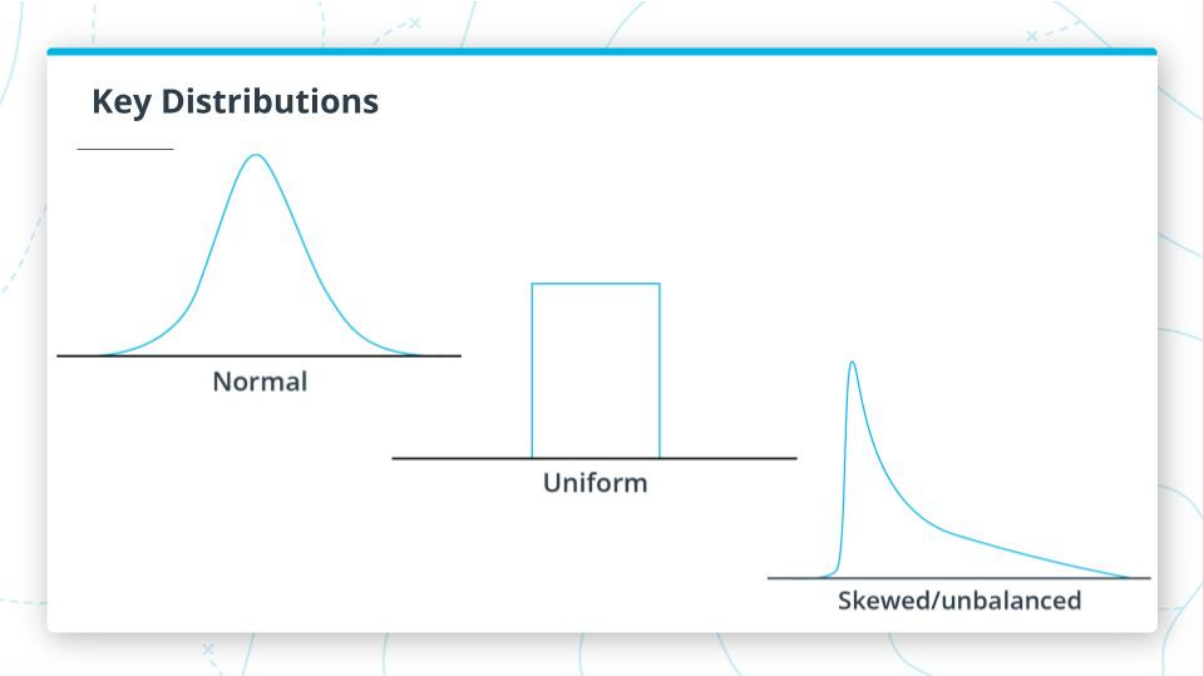
Key Things to Consider:

- 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.

Value Distributions



Key Distributions

Value Distributions Review

- Normal is the well-known bell-curve that most people are familiar with and is also referred to as a Gaussian distribution
- The uniform distribution is where the unique values have almost the same frequency and this is important b/c this might indicate some issue with the data
- Skewed/unbalanced data distributions as the name indicates are where a smaller subset of values or a single value dominates
- Bimodal and Poisson but for the scope of this course we will not cover those

Missing Values and Outliers

Missing values are especially common in healthcare where you may have incomplete records or some fields are sparsely populated

Missing Data Classification

MCAR which stands for Missing Completely at Random. This means that the data is missing due to something unrelated to the data and there is **no systematic reason** for the missing data. In other words, there is an equal probability that data is missing for all cases. This is

often due to some instrumentation like a broken instrument or process issue where some of the data is randomly missing.

MAR refers to Missing at Random and this is the opposite case where there **is some systematic relationship** between data and the probability of missing data. For example, there might be some missing demographics choices in surveys.

MNAR is a Missing Not at Random and this usually means there is a relationship between a value in the dataset and the missing values.

Understanding why data is missing help with choosing the best imputing method to fill or drop the values in your dataset.

Code Concepts

Create a function to check the percent of missing and zero values you have.

```
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
```

Apply that function to the original dataframe `check_for_missing_and_null(dataframe)`

View the results and see if there are any values that stand out. Again you may need to deal with different columns in different ways depending on their type and reason for missing or zero values.

Additional Resources

- **Imputation Methods**
- **Advanced Imputation Methods**

Analyzing Dataset for High Cardinality

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?

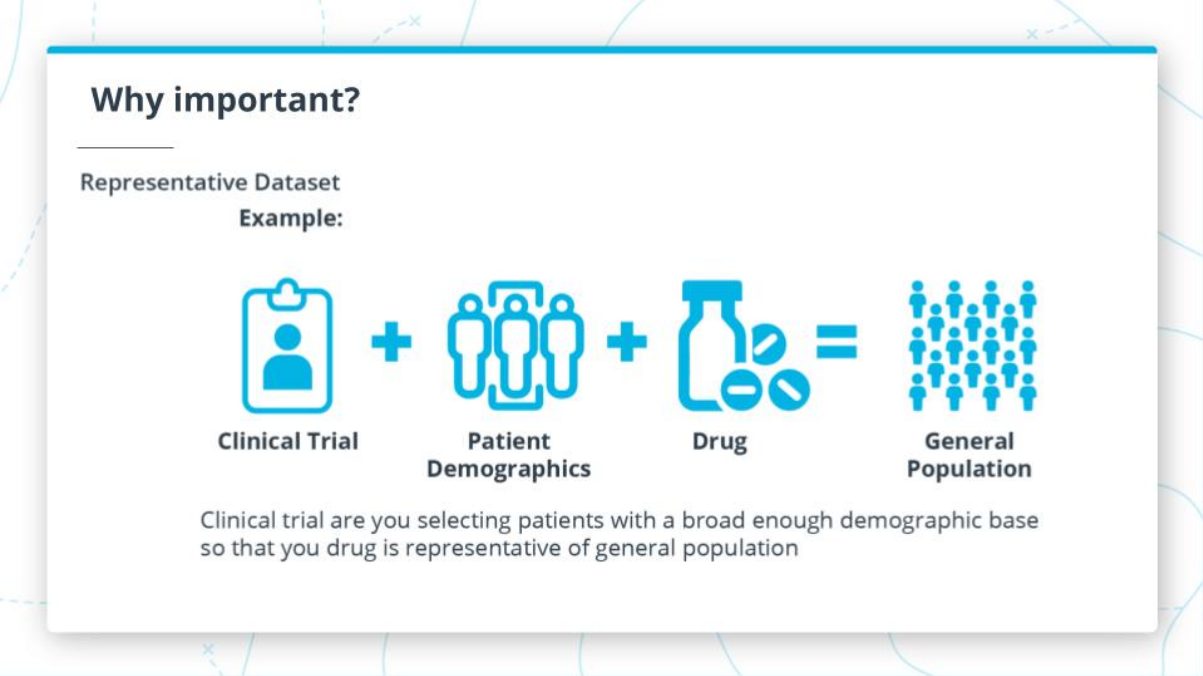
- Determine if it is a categorical feature.
- Determine if it has a high number of unique values. This can be a bit subjective but we can probably agree that for a field with 2 unique values would not have high cardinality whereas a field like diagnosis codes might have tens of thousands of unique values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

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

Additional Resources

Reducing Dimensionality

Demographic Analysis



The reason that demographic analysis is so important, especially in healthcare, is that we need our clinical trials and machine learning models to be able to representative to general population. While this is not always completely possible given limited trials and very rare conditions it is something we need to strive for and identify as early as possible if there may be an issue.

If we don't have a properly representative demographic dataset, we wouldn't know how a drug or prediction might impact a certain age, race or gender which could lead to significant issues for those not represented.

When completing a demographic analysis, it can be helpful to group data into buckets or bins.

In this walkthrough, we used `np.arange()` to create the bucket ranges, then used them to create the applies with a `.join()` method, and finally used `pd.cut()` to create our new "age_bins" buckets. You can use whatever methods you would like to complete this task. We also took the opportunity to change the sex or gender column from 0,1 to "male" and "female" to further breakdown our categories and demographics using `replace()`. Again this should all be a review, but we are just including here to be clear.

You may also use different age buckets/bins and see how the value distribution looks for those bins.

Additional Resources

Demographic Analysis

EHR Data Security and Analysis Lesson Recap

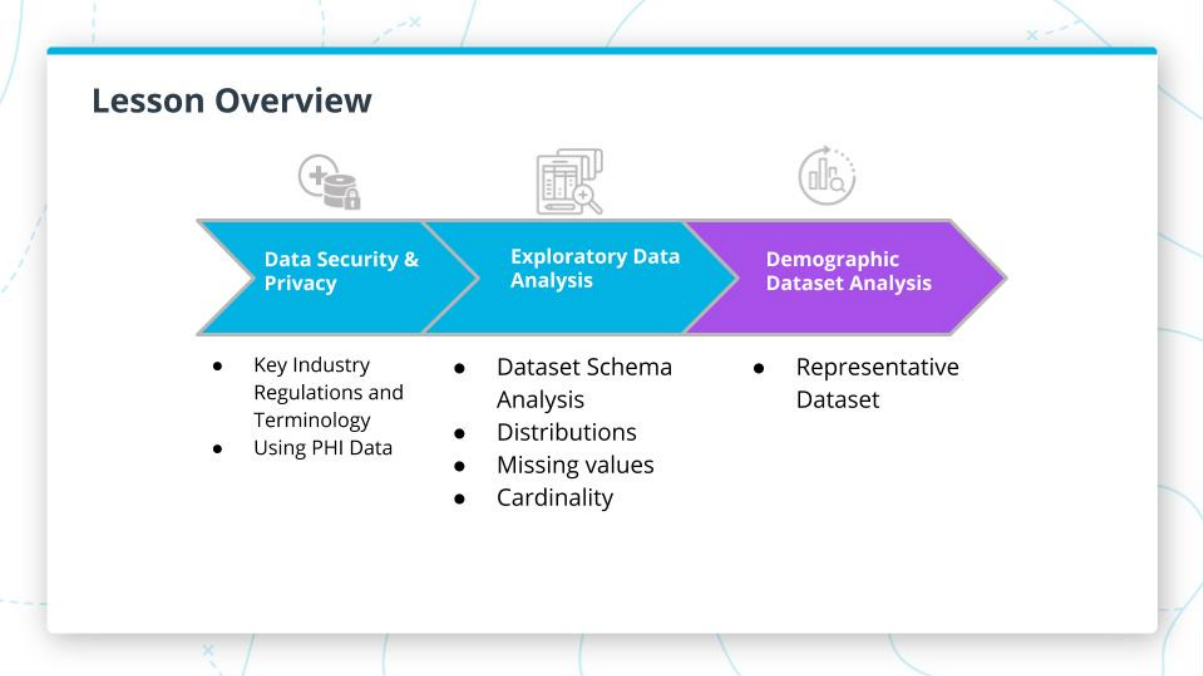
You have made it to the end of EHR Data Security & Analysis! Well done. You now know about Data Security and Privacy including some of the key standards and regulations. You have gained key information that you need to know about working with EHR data that will protect your data and you.

You also explored your way through data allowing you to gain a deeper understanding of your datasets including analyzing a dataset schema, looking at value distributions, missing values, and the cardinality of categorical features. It was a great adventure, wasn't it? I hope you took some pictures! Oh, wait maybe that wouldn't be a great data security practice.

And finally, you completed a demographic dataset analysis. You can now make sure that your dataset is representative before it would become a problem.

Once again, Congratulations! You're be ready to work with real healthcare datasets. You are also able to ensure that your training and predictions of the model you build later are truly representative. The next lesson is filled with the intrigue and mysteries surrounding...EHR Code Sets. I bet you didn't guess that one. I'll see you in the next lesson where you'll learn all about dealing with Diagnosis, Procedure, and Medication Codes.

EHR Data Security and Analysis



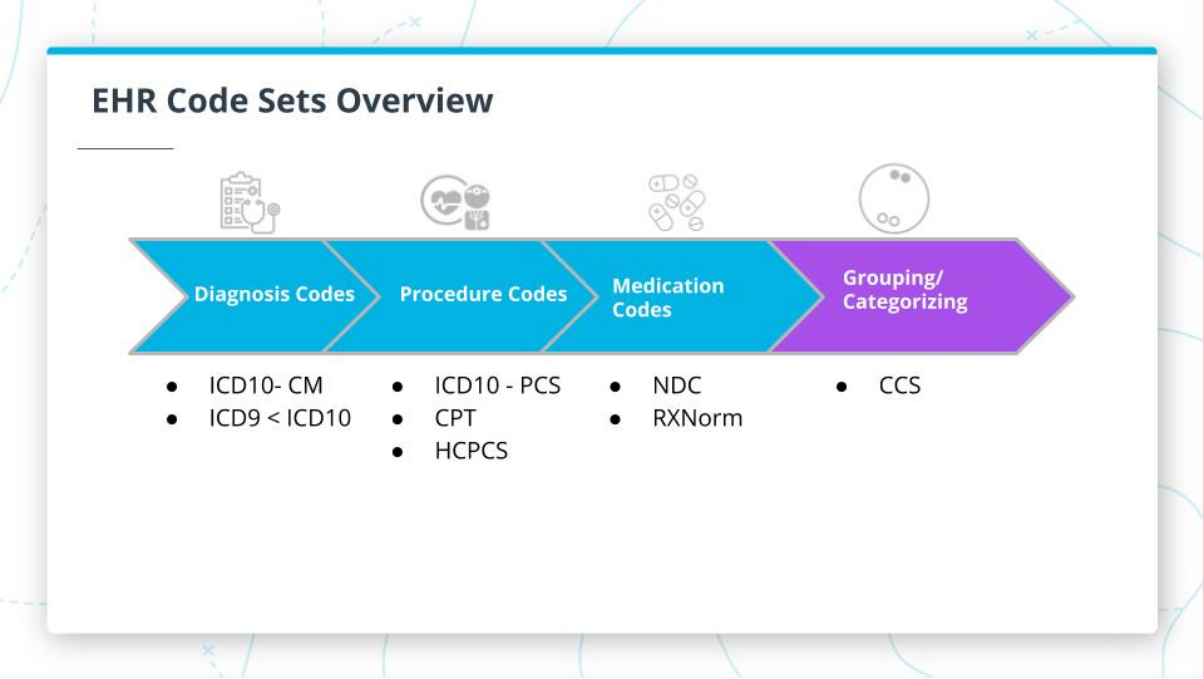
Lesson Key Terms

Aa Key Term	Definition

Key Term	Definition
Providers	These are entities/groups/individuals that provide care for patients. Providers can range in size from entire hospital networks to individual doctors and medical professionals.
Payers	a group that consists of companies like healthcare insurance. Payers can also include entities such as the government for things like Medicaid and Medicare in the United States.
HIPAA	The Health Insurance Portability and Accountability Act is the key industry regulation that you should be familiar within the U.S.
HITECH	The Health Information Technology for Economic and Clinical Health Act is also important to note and this is really just an update to HIPAA that accounts for technology.
EU	European Union
GDPR	The General Data Protection Regulation is generally considered more stringent than even HIPAA when it comes to protections for patients.
DPA	The Data Protection Act really builds off of and add to GDPR
PHI	Protected Health Information
Covered Entities	are a group of industry organizations defined by HIPAA to be one of three groups: health insurance plans, providers, or clearinghouses. You can see from the table the types of entities in each category.
Business Associates	A business associate is a person or entity that performs certain functions or activities that involve the use or disclosure of protected health information on behalf of or provides services to, a covered entity.
Business Associates Agreement/Addendum (BAA)	The contract between a covered entity and business associate.
De-identifying a Dataset	The removal of identifying fields like name, address from a dataset. De-Identification is done to reduce privacy risks to individuals and support the secondary use of data for research and such.
Expert Determination Method	Completed by a statistician to determines there is a small enough risk that an individual could be identified.
Safe Harbor	The removal of 18 identifiers like name, zip code, etc.
EDA	Exploratory Data Analysis
CRISP-DM	This stands for “cross-industry standard process for data mining” and is a common framework used for data science projects and includes a series of steps from business understanding to deployment.
MCAR	Missing Completely at Random. This means that the data is missing due to something unrelated to the data and there is no systematic reason for the missing data.
MAR	Missing at Random and this is the opposite case where there is some systematic relationship between data and the probability of missing data.
MNAR	Missing Not at Random and this usually means there is a relationship between a value in the dataset and the missing values.
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.

EHR Code Sets

EHR Code Sets Overview



You are about 20% of the way through and you have gained some very important skills in dealing with EHR data. Now you'll build even more on those skills as we begin to look at Electronic Health Record Code sets. But before we get started, let's review what will be covered in this lesson.

This lesson will delve into Code Sets in EHR and how to work with them, including:

- Diagnosis Codes
  - IDC10-CM
- Procedure Codes
  - ICD10 - PCS
  - CPT
  - HCPCS
- Medication Codes
  - NDC Codes
  - RXNorm
- Using these codes to group and categorize your datasets
  - Using CCS

This is very important because of how many codes there are in the medical field. If you don't understand the codes, you will likely not get all of and/or the right data you need for creating your models. So get ready for a lot of different codes! You'll be a master at figuring them out and using them to group your data by the time you're done.

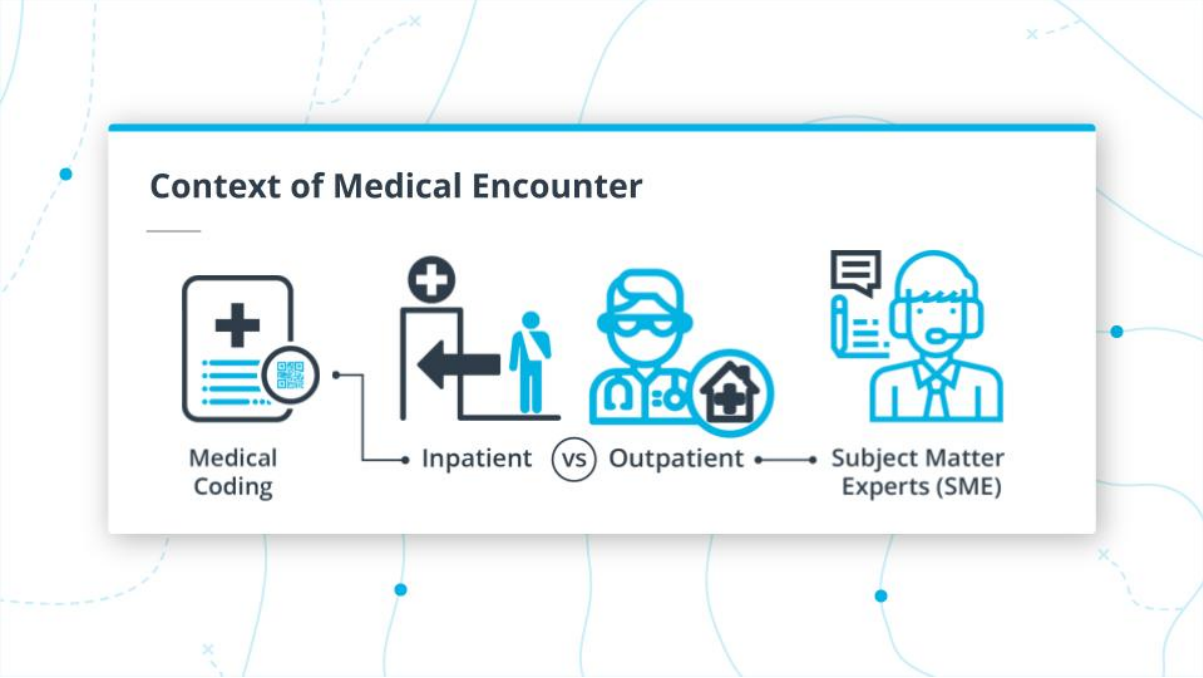
Codes Sets Background

For the purposes of this course, when we refer to a code set we are referencing an EHR data field that is linked to an accepted code standard such as ICD10 or CPT codes, which you will learn more about shortly. Examples of important code sets are diagnosis codes and procedure codes. We will go over these in more detail later.

Why do we need code sets?

There are many different providers and EHR systems around the world. There needs to be a standard way to encode common diagnoses, medications, procedures, and lab test results across all these providers and systems. We will focus on some of the most common code sets that allow for some of the most high-value analysis.

Context of a Medical Encounter



Context of a Medical Encounter

Medical Encounter: An interaction between a patient and healthcare provider(s) for the purpose of providing healthcare service(s) or assessing the health status of a patient.

This could also be online in today's world. During each encounter, one or more codes are generated about your encounter.

1. It is important to note that medical records often go through a process from a written note or some other unstructured input to a structured medical code that is standardized

but can be very specific. There is a whole industry of medical coders that take this unstructured information and “code” it to medical code standards and into EHR records.

2. Another piece of information that is relevant to industry codes is whether the encounter occurred in an inpatient or outpatient setting.
    - Inpatient usually refers to hospitalization.
    - Outpatient refers to visits and encounters like visits with your primary care physician or specialists like a cardiologist but do not require hospitalization.
    - Outpatient can also refer to ambulatory care.

It is important to work with subject-matter experts who know the domain and processes at a much deeper level. There are a lot of intricacies and details in healthcare that require you to work deeply with SMEs to understand the context better.

However, you should feel empowered to bring the technology and AI perspective to the table and re-think legacy processes!

Importance of Using Codes to Group/Categorize Your Data

As you will see in a bit there are literally thousands of medical codes in use and each time have a **Medical Encounter**, you have several of these codes added to that encounter. If you do not properly group/categorize your data, you may end up with missing or incorrect data to feed into your model. As an example let's say you knew one of the diagnosis codes for Sepsis and wanted to build a model around predicting which patients are at the greatest risk for Sepsis. Sepsis- sepsis during labor (O75.3) If you used only that code to build your dataset for training you are likely missing out on thousands of other records that also deal with Sepsis, but have a different code.

Here is a link to the Sepsis codes to see some others: **IDC10-CM Sepsis Codes**

This would lead your model to be inaccurate. This concept will become clearer as dig deeper.

Diagnosis Codes Part 1

Diagnosis Codes Key Points

In Healthcare, the different diagnosis code sets are extremely important and can sometimes be tricky to deal with. But don't worry, when you are done with this section, you'll be a able to crack the code like a pro!

What is a diagnosis?

Let’s imagine that you see a doctor and you have some symptoms such as coughing and a stomach ache and then your doctor magically comes up with a diagnosis. As we will learn later this diagnosis is a key piece of information that connects so much of the encounter experience together.

ICD10

ICD10: International Classification of Diseases 10

- Also known as ICD10
- Standard developed by **WHO**: World Health Organization and in 10th revision

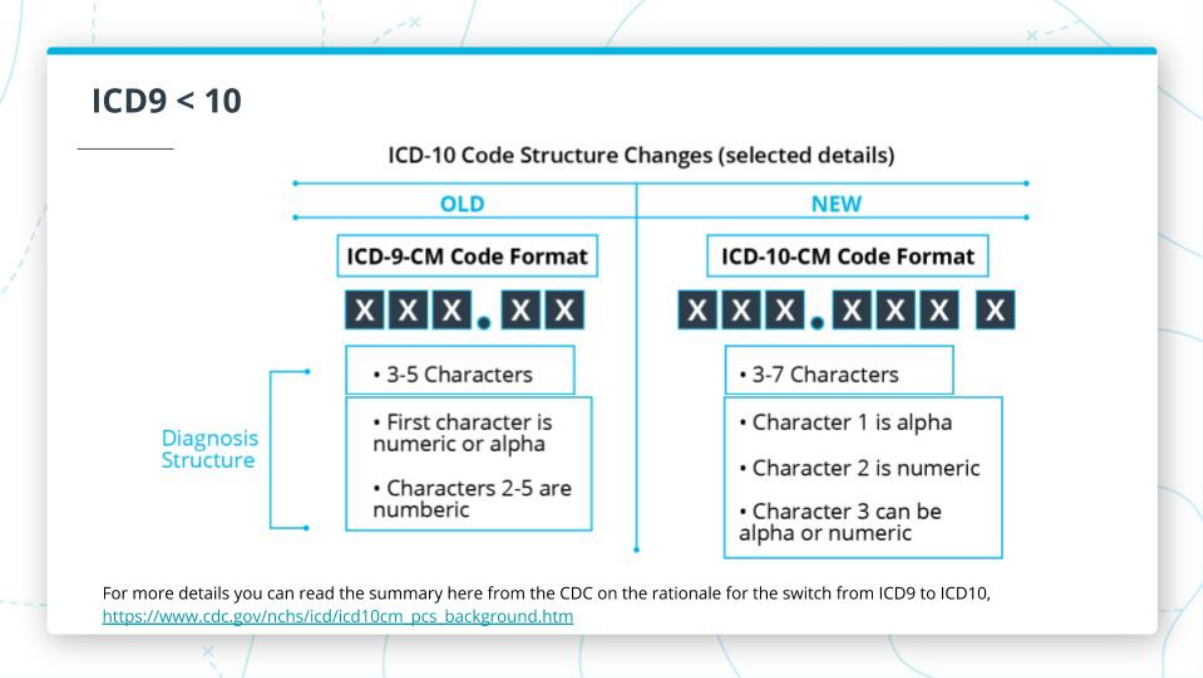
ICD10-CM

ICD10-CM: International Classification of Diseases 10 - Clinical Modification

Diagnosis code standard used in the U.S.

- Maintained by U.S. CDC
- Contains a wide variety of diseases and conditions
- Used for Medical claims, disease epidemic, and mortality tracking

ICD9 to 10



ICD9 to 10

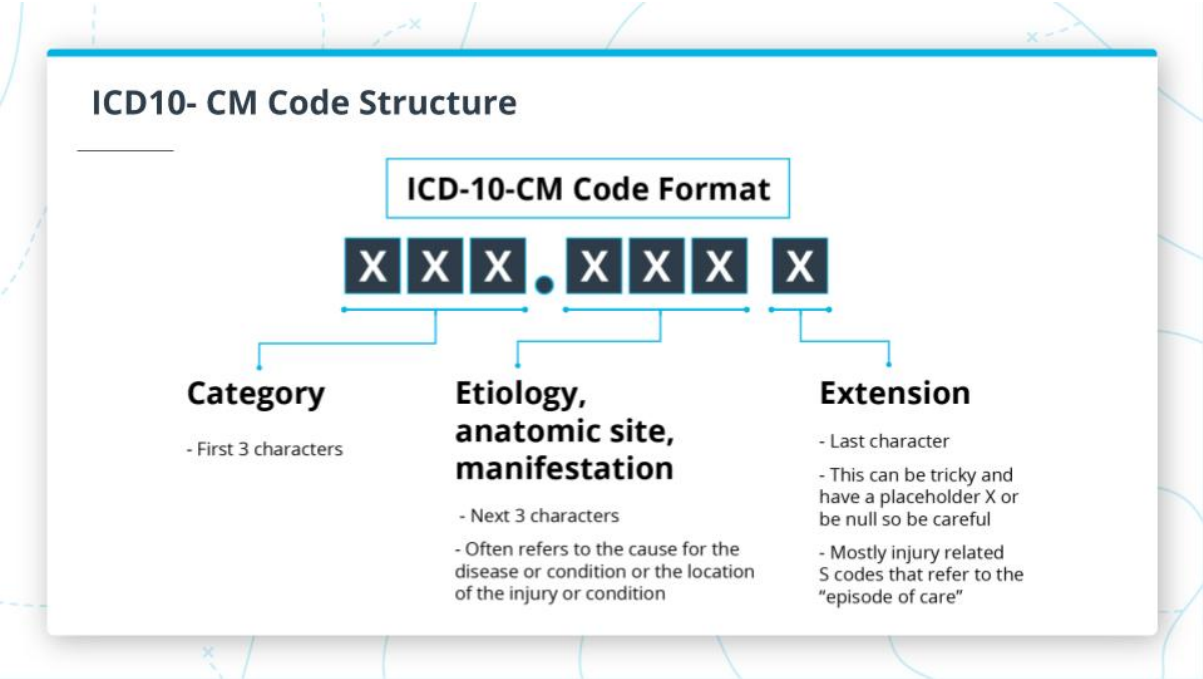
The U.S. had been using ICD9 since 1979 until it adopted ICD10 in October of 2014. The US lagged a bit behind the rest of the world who had adopted ICD10 much earlier. The reason for the changes from ICD9 to ICD10 is that the code set was not robust enough to meet future healthcare needs.

Important Changes:

- 14,025 codes ⇒ 69,823 codes
- Up to 7 characters
- Created a much higher level of detail
    - Included things like laterality or side of injury

You can learn more by looking at the **CDC's Rationale**

ICD10- CM Code Structure



ICD10- CM Code Structure

[XXX].XXX.X

The first part is the category of the diagnosis and it is the first three characters and could be an S code like the injury category. There are 21 different categories and these range from the disease of the respiratory system to injury, poisoning, and certain other consequences of external causes.

XXX.*[XXX]*.X

The Second is the etiology, anatomic site, and manifestation part which can be up to 3 more characters and is essentially the cause for a condition or disease or the location of the condition.

XXX.XXX.*[X]*

Finally, the third part is the extension which is the last character and can be tricky b/c it can often be null or has an X placeholder. It is often used with injury-related codes referring to the episode of care.

Additional Resources

Code Structure **Coding Guidelines**

Diagnosis Codes Part 2

First, it is important to note that it can take several encounters for a patient to receive a diagnosis, and therefore a diagnosis code. There may be an initial appointment, some follow-up testing, and then another appointment before a diagnosis is determined. This is very common. This means that there may not be diagnosis codes for every encounter.

Encounter 1: G30.9, M06.9, ~~**G30.9**~~

Second, a diagnosis code should never be repeated in the same encounter.

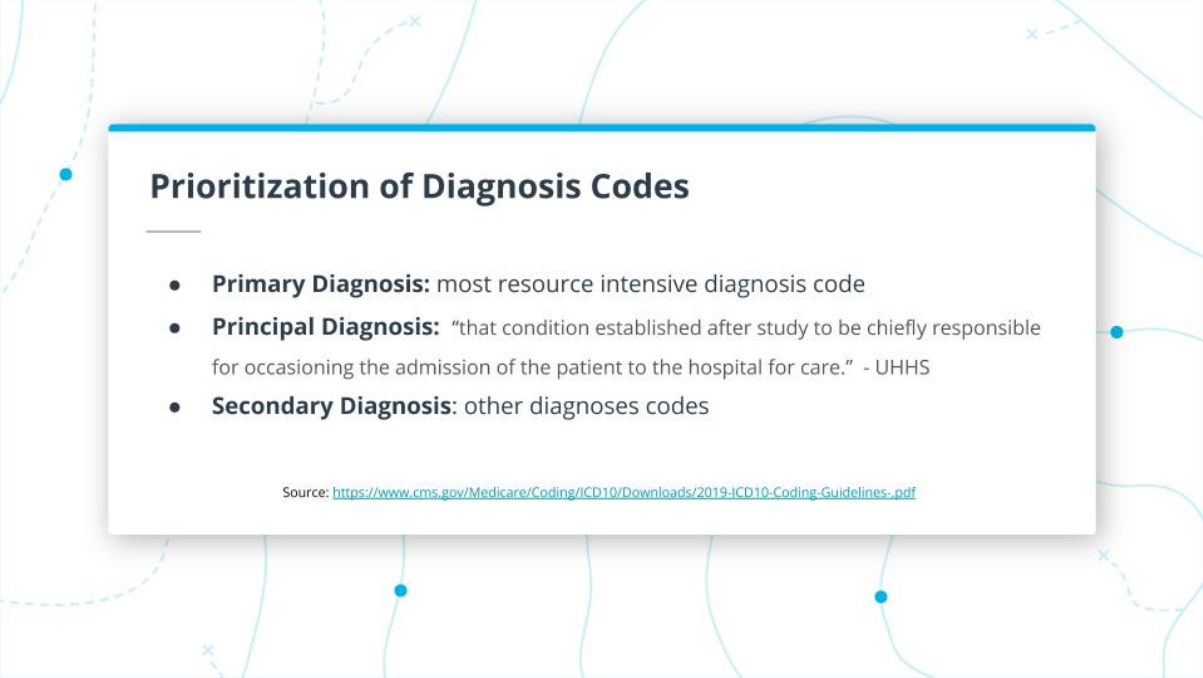
- Encounter 1: **G30.9**, M06.9
- Encounter 2: **G30.9**, M06.9, A34.6

Third, it is expected that a diagnosis would carry across different encounters as the patient is being treated for that diagnosis.

Additional Resources

In the video, we showed you the CDC lookup tool for ICD10-CM codes. The link is below. Go ahead and look up some codes you might be familiar with!

- **CDC ICD10-CM Lookup Tool**
- **Coronavirus ICD Code Announcement**



Diagnosis Code Prioritization

At a high level, it is important to distinguish what code is taking up the most resources or is the most critical and there are few terms that you should become familiar with.

- **Primary Diagnosis Code**: The code that takes up the most resources to treat.
- **Principal Diagnosis Code**: The diagnosis that is found after hospitalization to be the one that is chiefly responsible.

This **can be** an important distinction since the admitting diagnosis code can widely differ from the final, Principal Diagnosis. For the most part, these terms interchangeably but it's good to be aware of the differences and the need to dig into the details when necessary.

- **Secondary Diagnosis Codes**: The other diagnosis codes listed on an encounter.

For example, if a patient were to have a knee replacement surgery but had type 2 diabetes as a prior condition, the secondary diagnosis code of type 2 diabetes would be included in the medical record.

Note: Secondary diagnoses codes can include many additional codes

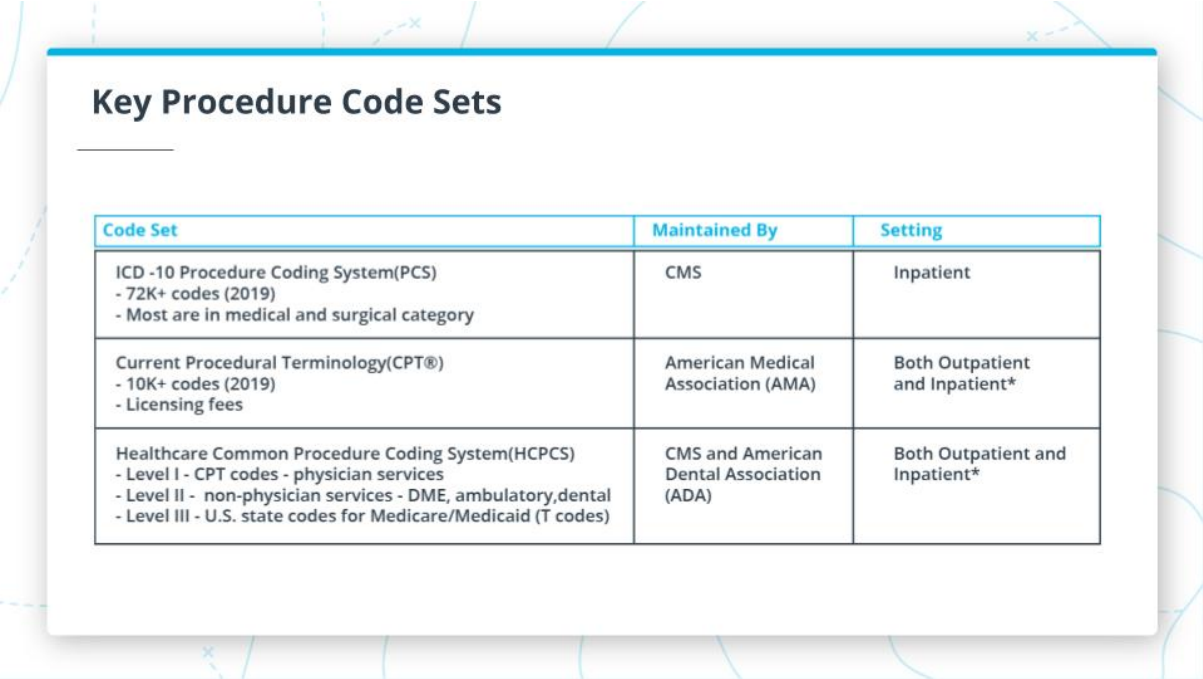
Additional Resources

- **ICD10 -CM Coding Guidelines**
- **Primary, Principal, and Secondary Diagnosis Codes**

Procedure Codes

Procedure Codes: the categorization of the medical codes during an encounter. It's important to note if a procedure is inpatient or outpatient.

Key Procedure Code Sets



Key Procedure Code Sets

The graphic above shows some additional information about 3 of the important code sets. Here are the key points about each of them.

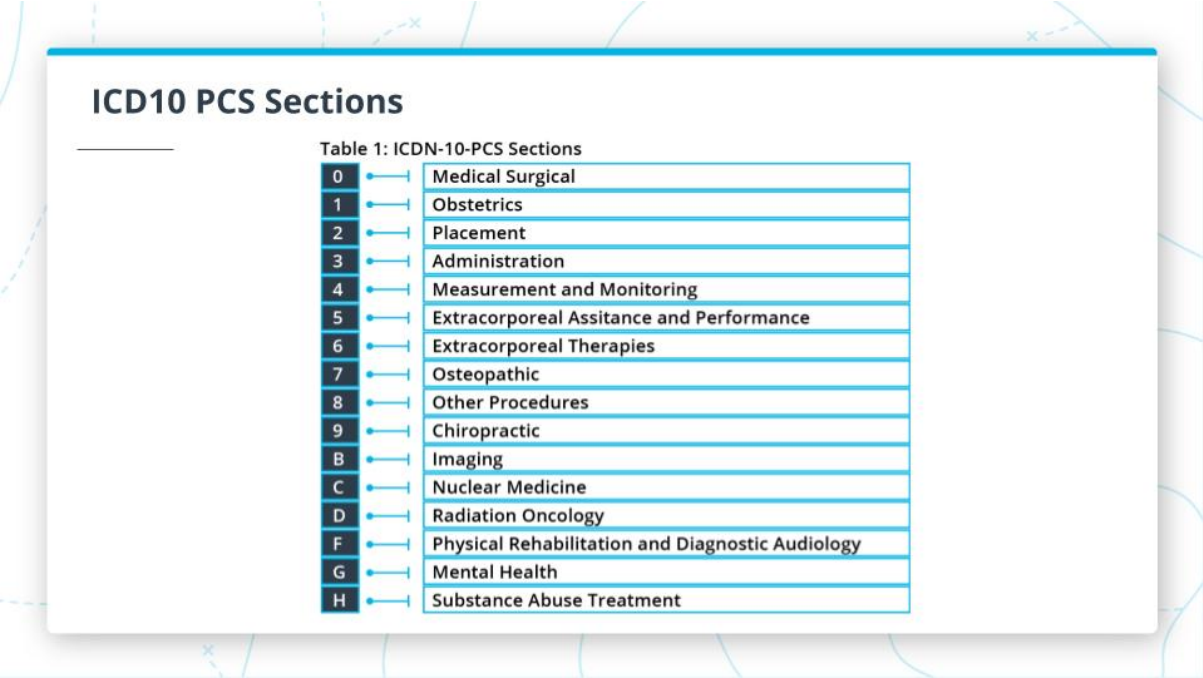
- **ICD10 PCS**: Procedure Coding Systems
  - Only for **Inpatient**
  - 72,000+ codes as of 2019
  - Focus on medical and surgical
- **CPT**: Current Procedural Terminology
  - **Outpatient** focused but can apply to physician visits in ambulatory settings

- 10,000+ codes as of 2019
- Focus on professional services by physician

- **HCPCS**: Healthcare Common Procedure Coding System
  - Inpatient and outpatient
  - Has 3 levels
    - L1: CPT Codes
    - L2: Non-physician services
      - DME: Durable Medical Equipment
      - Ambulatory Services
      - Dental
    - L3: Medicare/Medicaid related

We will focus on ICD-10 PCS and CPT codes, but it's good to be aware of HCPCS codes, too!

ICD10 PCS Selections



ICD10-PCS Code Structure

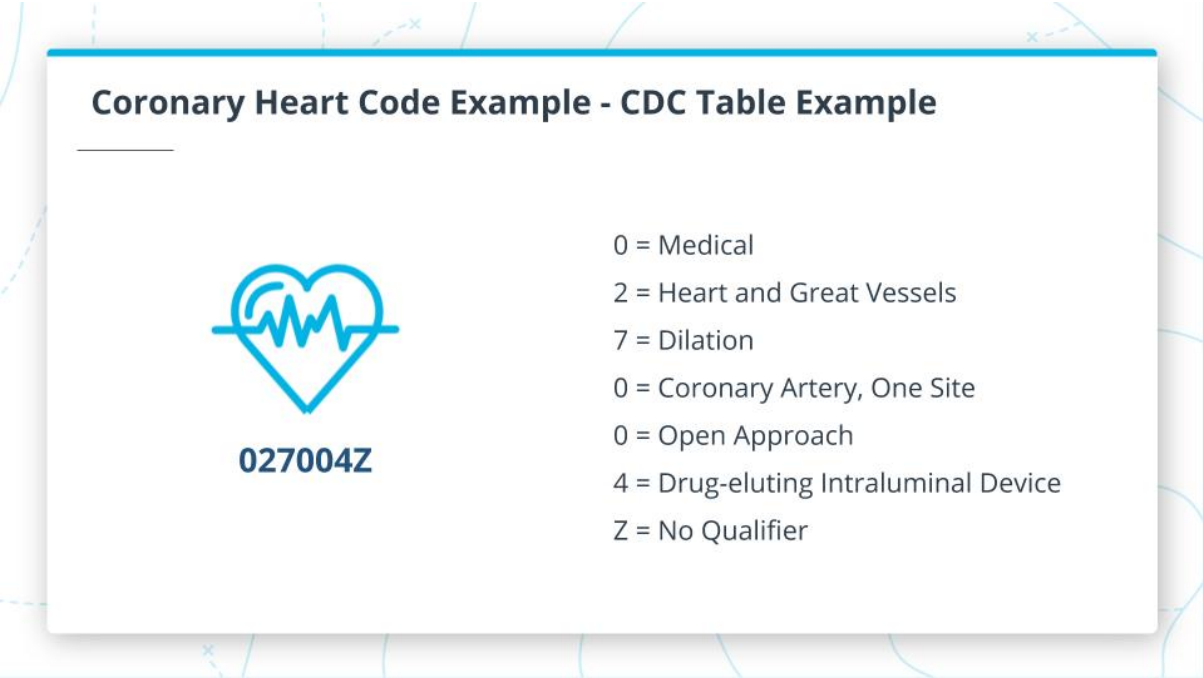
- 7 alphanumeric characters [A-Z, 0-9]
- 1st character is the Section
  - Reference Section codes categories in the table above
- Subsequent characters relate to the *Section* and give:
  - Body System
  - Body Part

- Approach
- Device used for a procedure

Additional Resources

- **PCS Procedure Code System**

Coronary Heart Code Example: 027004Z



PCS Code Example

For this example we used 027004Z. This is a Heart Surgery code that relates to:

- Dilation of Coronary Artery
- One Site with Drug-eluting Intraluminal Device
- Open Approach

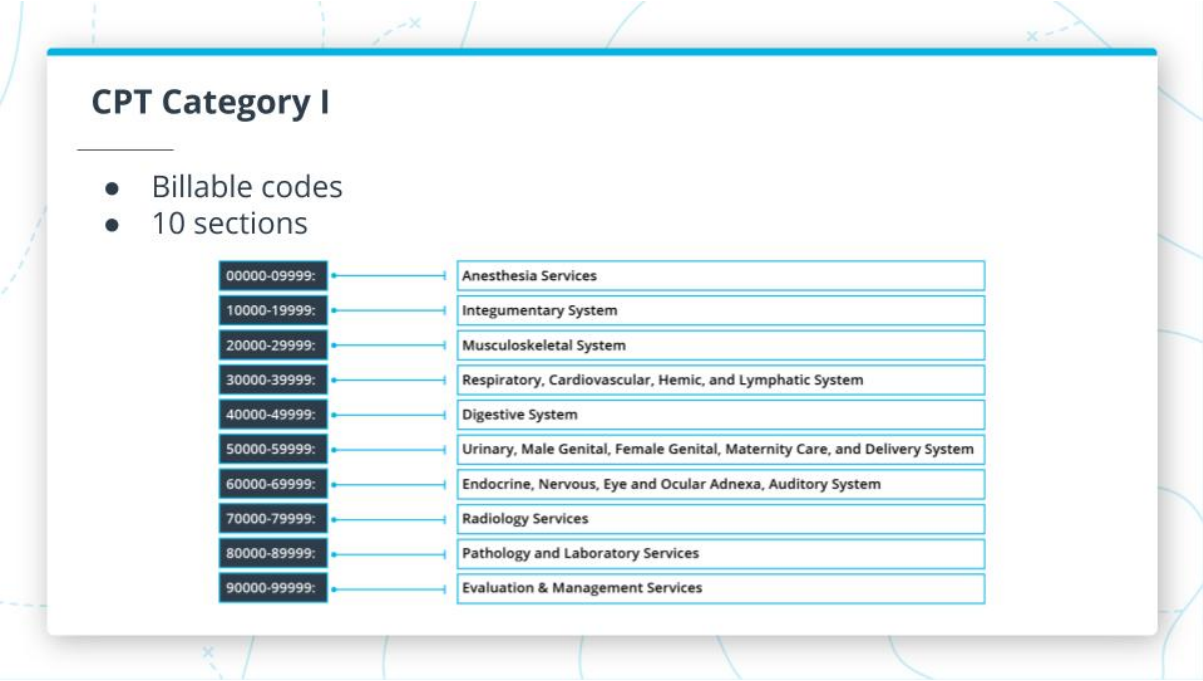
You can see the breakdown in the image above. In the table, you can see that this code starts with **0**, which is the **Medical** category. If you look over the link from the last section, **PCS Procedure Code System**, you can further see where the rest of the code breaks down.

- 0 = Medical
- 2 = Heart and Great Vessels
- 7 = Dilation
- 0 = Coronary Artery, One Site
- 0 = Open Approach
- 4 = Drug-eluting Intraluminal Device
- Z = No Qualifier

Additional Resources

- **PCS Overview**
- **Medicare Coding**

CPT Codes



CPT Category 1 Billable Codes

CPT codes have the following structure:

- Up to 5 numbers
- 3 Categories
  - Category 1: Billable codes
    - 10 sections largely split along biological systems which are broken out in the image above
  - Category 2:
    - Five digits that end in F
    - Non-billable
    - Performance measure focused on physicals and patient history
  - Category 3:
    - Services and procedures using emerging technology

Generally, you'll focus on the first two categories.

As a reminder, we will not be covering HCPCS Codes. However, we have added some additional links regarding HCPCS codes below.

Additional Links

- **Understanding CPT Codes**
- **CMS.gov Database Download for CPT Codes**
- **CMS.gov HCPCS Codes Information**
- **HCPCS Wikipedia**

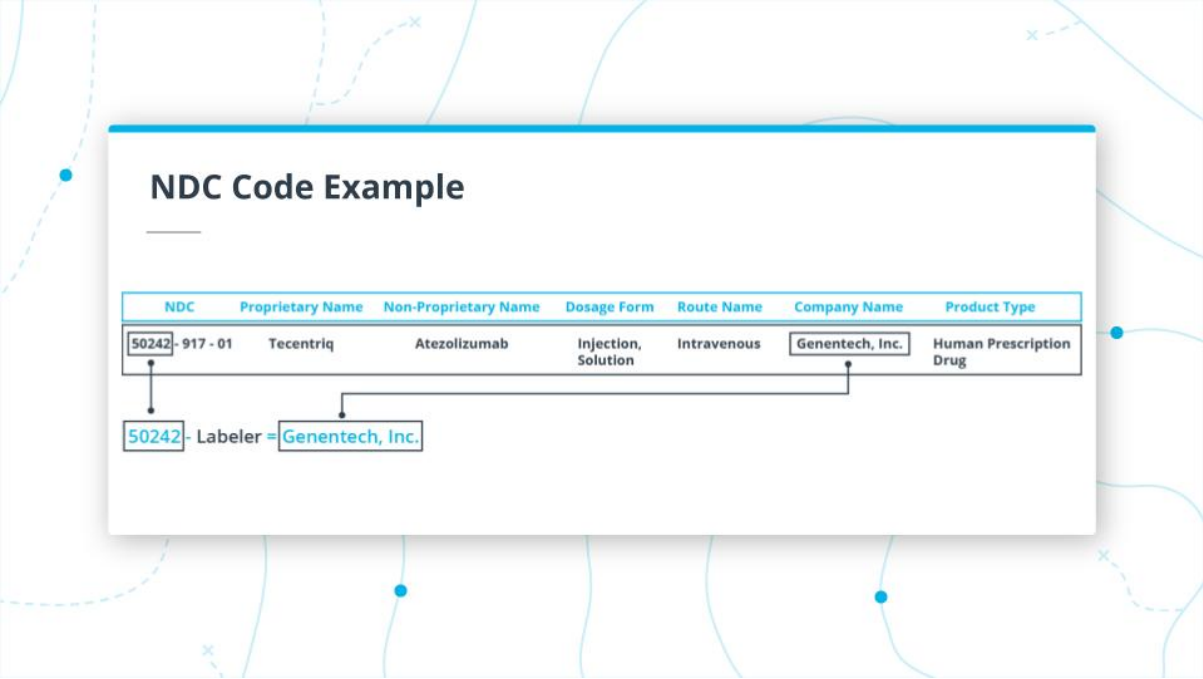
Medication Codes

NDC: National Drug Code

In this section, we discussed the NDC Codes. These codes have been in place since 1972, and are maintained by the FDA.

NDC Code Structure

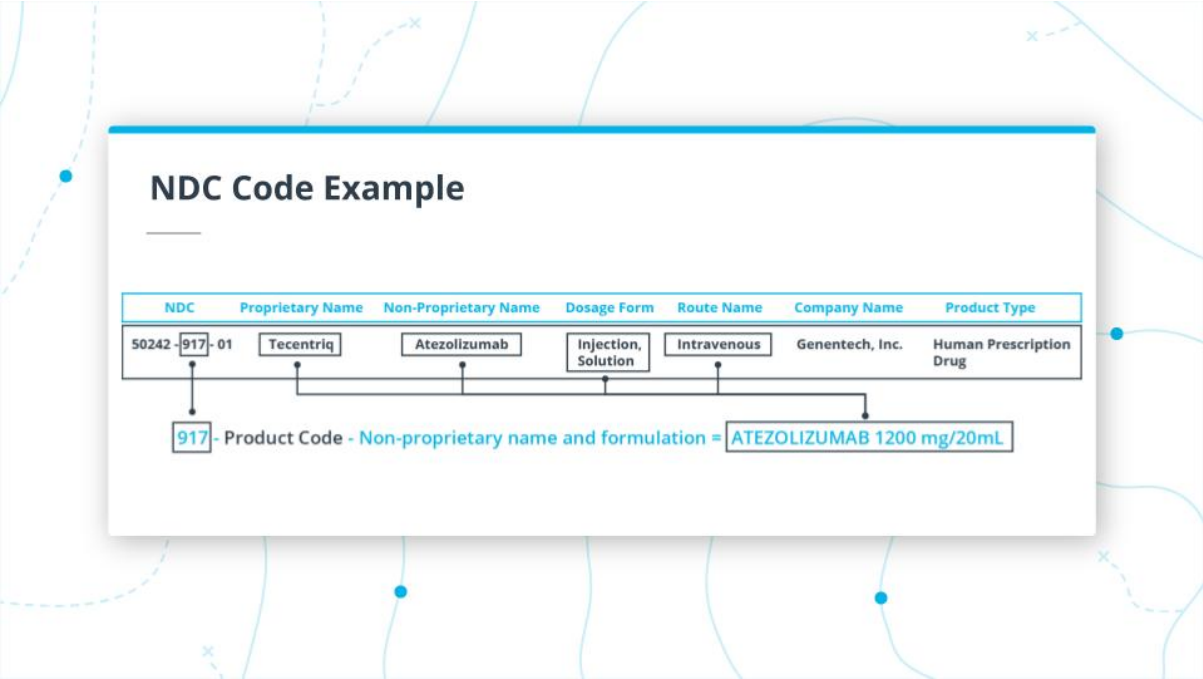
- 10- to 11-digit code
  - multiple configurations
- 3 Parts
  - Labeler: Drug manufacturer
  - Product code: the actual drug details
  - Package code: form and size of medication



In the image above you, can see the first part of the NDC code for *Tecentriq*.

The first part of the code **50242** is the **Labeler**, which maps to the manufacturer of the drug (which in this case is **Genentech, Inc**).

NDC Product Code

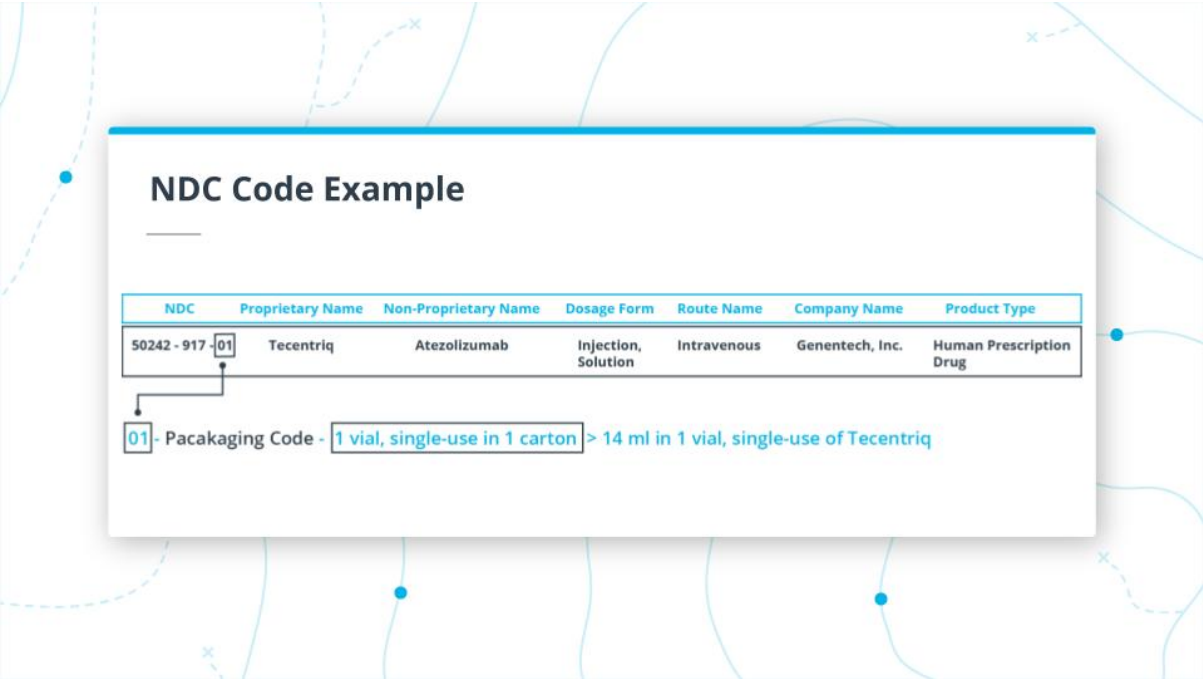


In the example above, we are looking at the middle section of the code **917**.

917 is the **Product Code**. In this case, it takes the unpronounceable, Non-Proprietary Name, **ATEZOLIZUMAB 1200mg/20ml** and maps it to the Proprietary Name, **Tecentriq**. It also indicates that the drug dosage form is **Injection, Solution** and the route of administration is **Intravenous**.

It is important to note that you may need to use the Non-Proprietary Name in order to group a drug by all of its generics.

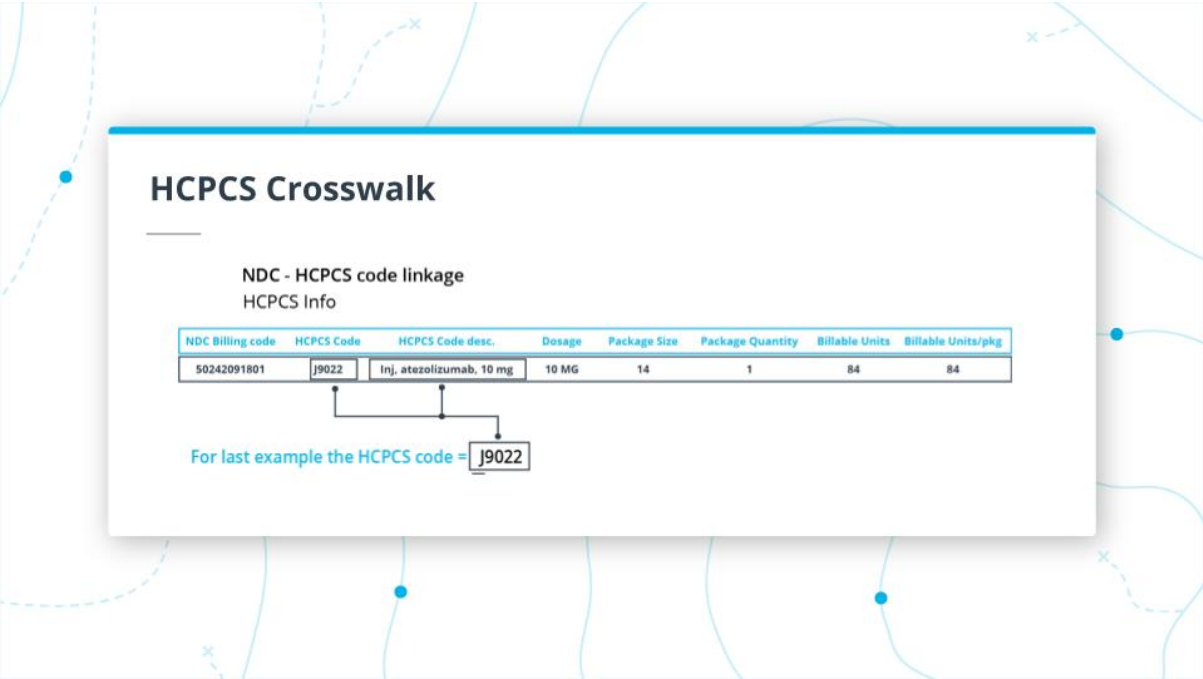
NDC Packaging Code



Finally, the last two digits **01**.

This is the **packaging code**, and in this instance indicates that the package is a 14ml single vial in 1 carton for single use.

HCPCS Crosswalk



Crosswalk

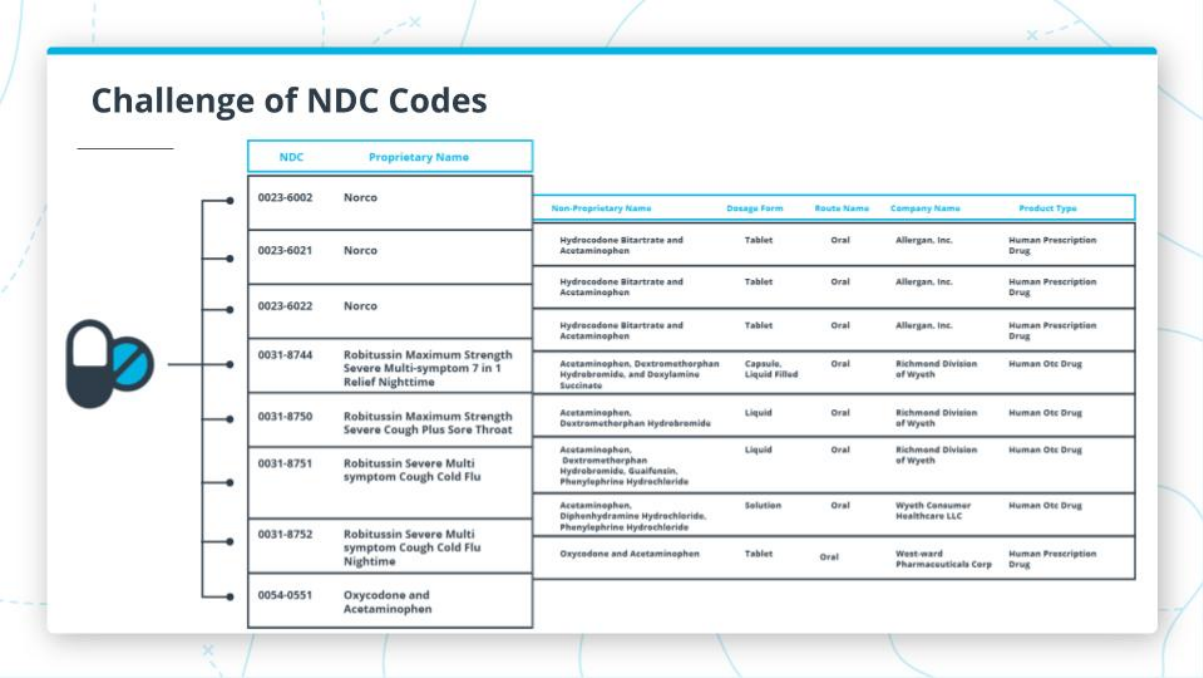
Crosswalk: A connection between two different code sets or versions of drugs in the same code set.

A common term you will hear for medical code sets is a crosswalk.

For NDC codes we can connect them to HCPCS codes and depending on the data source you are looking at there may be a mapping from these codes.

For the previous example, there is a crosswalk between the Tecentriq NDC code and the HCPCS code J9022. Something to note is that the HCPCS code starts with a J and the “J” codes tend to be drugs that are injected but by definition, they are drugs that are not taken orally. You can see that the HCPCS code maps some info from the NDC code.

NDC Code Challenges



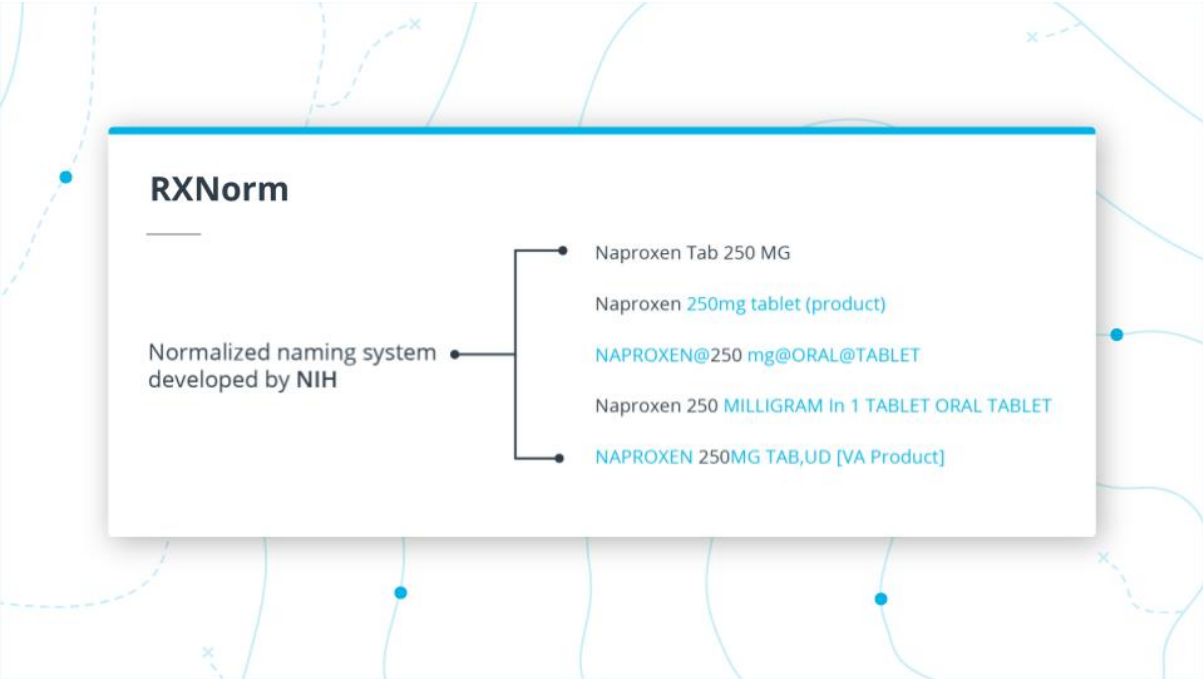
Challenge with NDC Codes

One of the major challenges with NDC codes is to normalize or group codes by common types of drugs. For example, let’s take acetaminophen. You would think this would be relatively straightforward to group whenever someone has the NDC code for this drug. However, as you can see from this table of only a sample of the results from **NDC List** that there many drug codes that could contain acetaminophen with other drugs too. This can make drugs very difficult to deal with.

Additional Resources:

NDC List

RXNorm Example



RXNorm

To address the problem just mentioned, the NIH developed a normalized naming system called **RXNorm**, which does what its name implies and groups medication together. This is important because providers, pharmacies, payers all send EHR records with this data but might use different names and it becomes difficult to communicate between different systems.

To illustrate this issue, take a look at a drug Naproxen and the just a few examples of different names of naproxen that is the same thing,

While there is a crosswalk between NDC codes and RXNorm, there are still some issues. Depending on the system you are dealing with, it could use one or the other code set.

Additional Resources

RXNorm Overview

Grouping/Categorizing Systems

CCS- Clinical Classifications Software

As mentioned earlier, there is a tremendous challenge of taking the 77K+ ICD10 PCS codes and categorizing them into meaningful categories at scale. This is where a government-industry partnership called the Healthcare Cost and Utilization Project (HCUP) created a categorization system called clinical classifications software or CCS. It can be used to map diagnosis or procedure codes from ICD code sets. It has single or multi-level options for mapping these codes.

Single Level Categories

- Mutually exclusive categories

- 285 categories for diagnoses
- 231 categories for procedures

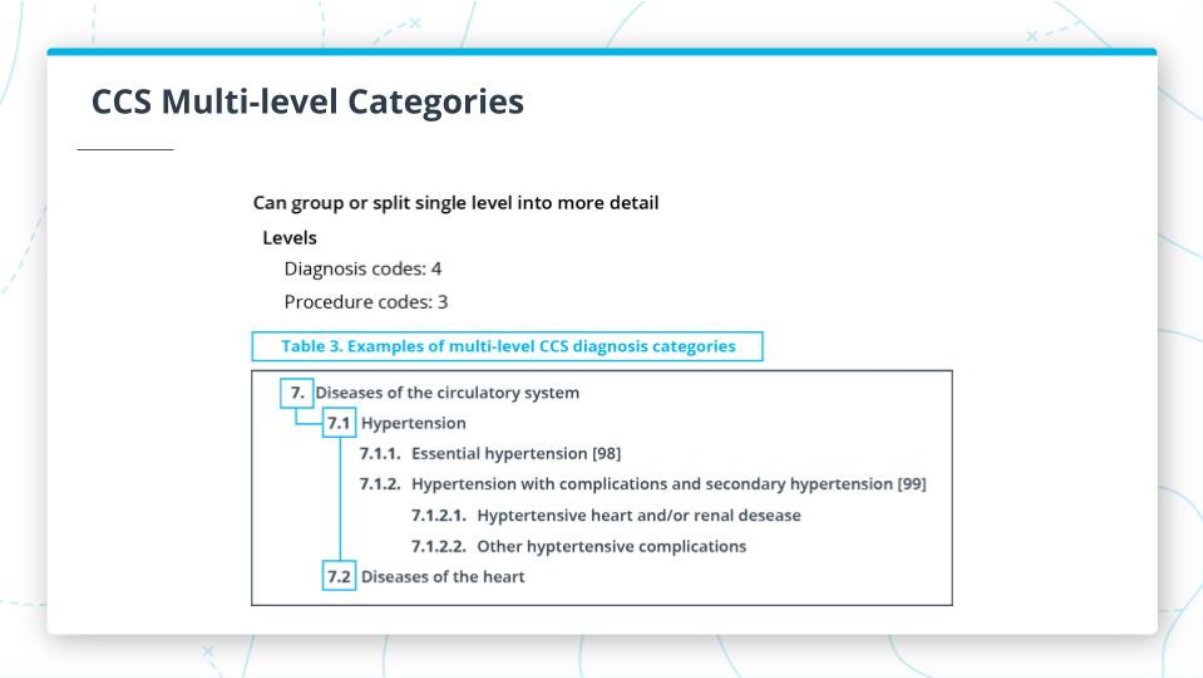
As an example using single-level codes:

- **Operations on the cardiovascular system** are codes **43-63**
- **Heart valve procedures** is code **43**.

Below you can see how that changes using multi-level codes.

Single level categories can be used for ranking of codes and help with risk adjustment scoring

CCS Multi-level Categories



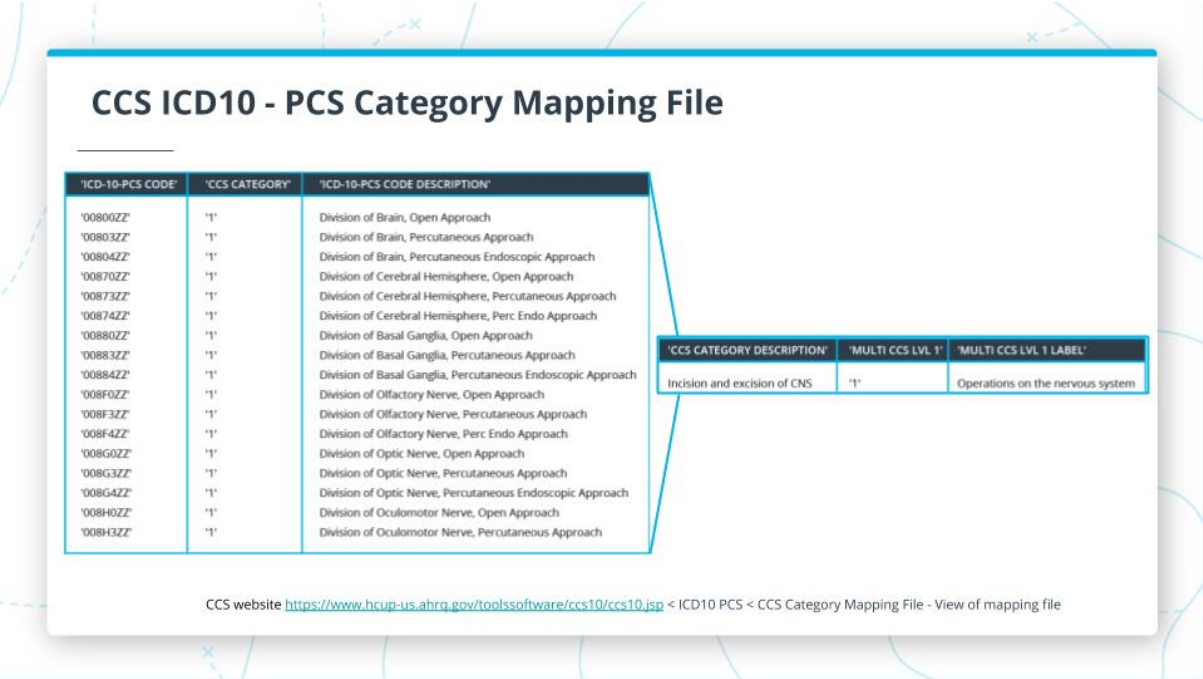
Multi-Level Categories

CCS Multi-level categories are helpful for more detailed analysis and grouping at a more granular level. there are 4 levels for diagnosis codes and 3 levels for procedure codes.

In the above example, **7** is a broad category, but then you can see that it branches off into **7.1** and **7.2** into more detail. **7.1**.

Then **7.1.X.X** further break down hypertension.

7.1.2.1 = Hypertensive heart and/or renal disease (*typo in image above*).



CCS ICD10 - PCS Category Mapping File CCS website

CCS ICD10-PCS Category Mapping File

Below we have included a link to the CCS ICD10-PCS Category Mapping File which makes figuring out and working with these category groupings much easier. *You download the mapping file for taking ICD10 PCS procedure codes and map them to grouping/categorization with CCS.*

The mapping file breaks down like this (image above):

- ICD10 codes and their descriptions on the left
- CCS category descriptions, Multi CCS LVL 1, and Multi Lvl 1 Label on the right.

It would be hard to have a medical expert categorize these at and they can change over time. Thankfully CCS maps these for us. These mappings can be helpful in reducing dimensionality.

Other Categorization Systems

- **MS- DRG**- Medicare Severity-Diagnosis Related Group
  - Group payment based on the principal diagnosis
  - Up to 25 secondary dx
  - Up to 25 procedures during a visit/encounter
- **SNOMED CT**- Systematized Nomenclature of Medicine—Clinical Terms
  - License to use
  - Helpful for making the EHR records interoperable

We will not be using these two in this course and there are many other systems as well, but knowing that they exist is good.

Additional Resources

- **CCS website**
- **MS-DRG**
- **SNOMED CT**

EHR Code Sets Recap

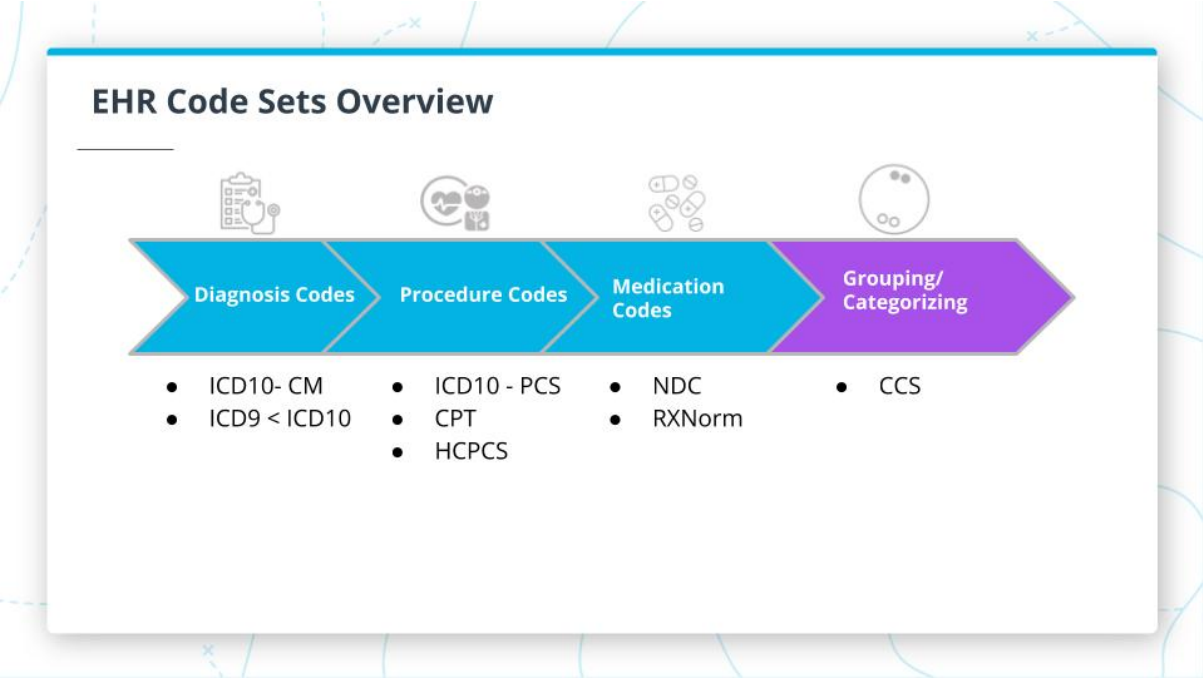
Awesome! You made it to the end of code sets in EHR.

In this lesson, you learned about key components of EHR data, Code Sets. As a quick reminder, EHR Code sets allow for the comparison of data across various EHR systems. You first learned about diagnosis codes. In particular, ICD10-CM. You gained knowledge of what this code set is, why it's important, how to read them. You also applied that knowledge by grouping data using EHR codes as well as build an embedding with visualizations. Hopefully, ICD10- CM codes will be easy to work with from now on.

Then you reduced cardinality of a dataset using procedure codes after learning more about ICD10-PCS, CPT and HCPCS. You should be able to break down medication codes too! You should also be able to apply your knowledge to deal with the challenges of working with medication code sets.

Finally, you put all of this code knowledge to work by grouping and categorizing these code sets with CCS. I know that there was a lot of information in this lesson, but now it's time to move on to learning some magic to transform EHR Data! Into what? You'll have to head to the next lesson to find out!

EHR Code Sets Overview



Key Terms

Key Terms	Definition
Medical Encounter	An interaction between a patient and healthcare provider(s) for the purpose of providing healthcare service(s) or assessing the health status of a patient.
ICD10:	International Classification of Diseases 10
WHO:	World Health Organization
ICD10-CM	International Classification of Diseases 10 - Clinical Modification
Primary Diagnosis Code	The code that takes up the most resources to treat.
Principal Diagnosis Code	The diagnosis that is found after hospitalization to be the one that is chiefly responsible.
Secondary Diagnosis Codes:	The other diagnosis codes listed on an encounter.
Medical Procedure	A course of action to achieve an intended result for a patient in healthcare.
Procedure Codes	The categorization of these medical codes during an encounter.
NDC	National Drug Code
Crosswalk	A connection between two different code sets or versions of drugs in the same code set.
RXNorm	Groups medications together.
CCS	Clinical Classifications Software used to map diagnosis or procedure codes from ICD code sets.
MS-DRG	Medicare Severity-Diagnosis Related Group
SNOMED CT	Systematized Nomenclature of Medicine—Clinical Terms

EHR Transformations & Feature Engineering

This lesson is divided into 3 parts:

1. 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.

1. Dataset Splitting Without Data Leakage

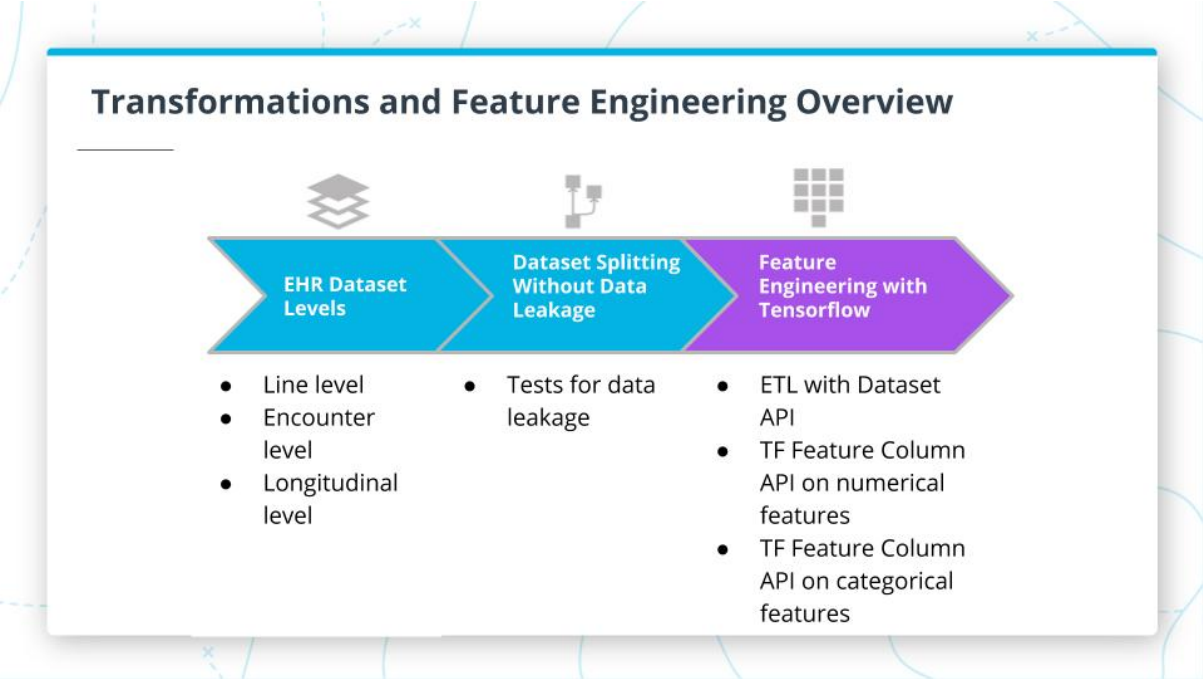
In this part, you will learn about dataset splitting without Data leakage, which can be a major issue in EHR datasets. By the end of part two, you will be able to implement some basic tests to help prevent issues when splitting data.

1. 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 API can be extremely useful for transforming datasets at scale and building some unique feature types.

Let's get started!



Transformations and Feature Engineering Overview

EHR Dataset Levels

With EHR datasets, there are three levels.

- Line
- Encounter
- Longitudinal.

These levels are extremely important in healthcare data, and being able to identify and work with data at the correct level will ensure that you start with the correct data type and dataset to feed to your models.

Line Level

Line Level: A denormalized or disaggregated representation of all the things that might happen in a medical visit or encounter.

Think of a visit to the doctor for bronchitis.

Your line-level data entries could be:

- A diagnosis code of bronchitis
- A medication code for a cough suppressant

- A procedure code for a test for bronchitis and a line could be a diagnosis or medication that was prescribed. Another line could include information on a lab test that the doctor ordered for informing the diagnosis.

Encounter Level

Encounter Level: Also known as the visit level, which is the aggregated information from the previously mentioned line level for one encounter. This information can be collapsed into a single row or arrays.

Using the example above, the encounter level for that visit would include the diagnosis code of bronchitis, medication code for a cough suppressant, and the procedure code for a test for bronchitis in one array or list.

Longitudinal Level

Longitudinal Level: Also known as the patient level view. This level aggregates the patient history and can show how the culmination of visits/encounter lead to some clinical impact.

Continuing with our example above if the patient contracts bronchitis often, over a series of years, we might gain some insights into a possible autoimmune disease or know exactly what to prescribe the patient when they start seeing symptoms.

Now that you have a basic understanding of the different levels we'll explore them a bit more with examples.

EHR Dataset Levels Continued

As stated above, EHR records are commonly represented at one of the following three levels: line, encounter, and longitudinal levels. Let's review this one more time with the visual above.

Patient A had an Encounter on January 20th of 2019, where they had 3 different codes produced. Patient A also had another encounter on March 20th of 2019 with its own set of codes. All together these encounters and line-level codes add up to the Longitudinal Level of knowledge we have on that particular patient.

The Longitudinal view is an important level for aggregating the patient history and is where you connect information across visits/encounters and rolls up information to determine trends across time.

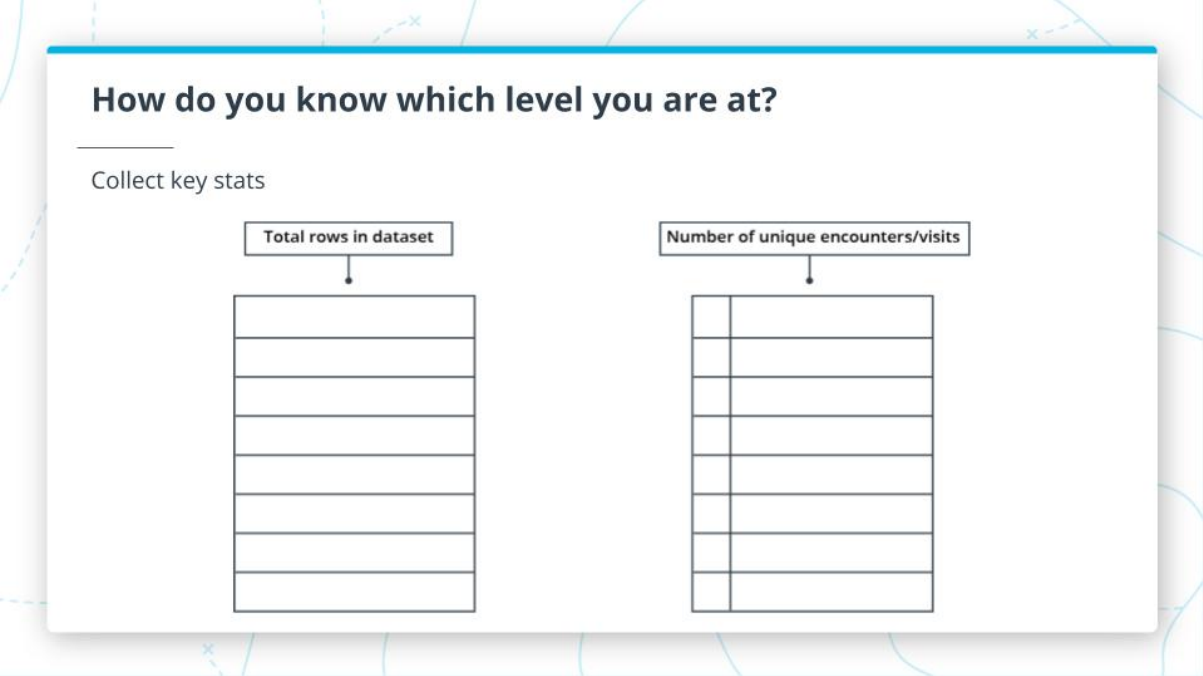
Why are EHR levels important?

Using the wrong EHR dataset level can lead to major errors with building models because data preparation is done with faulty assumptions and lead to serious error. For example, a common cause is the duplication of encounter information when you take a line-level dataset and treat it as an encounter level dataset.

Example:

A particular encounter might have 50 lines and that might be treated inadvertently as 50 distinct encounters when it is actually one encounter. This has the effect of upsampling certain common values for that encounter in your dataset, but also creates a great deal of noise since those 50 lines might have only slight differences.

Further, selecting the wrong encounter from the patient record can often occur and there might be a case where you only want the earliest or latest visit or state for a patient or time step for your model. This can cause many issues that might not become apparent until the modeling or deployment phases of your project



How do you know which level you are at?

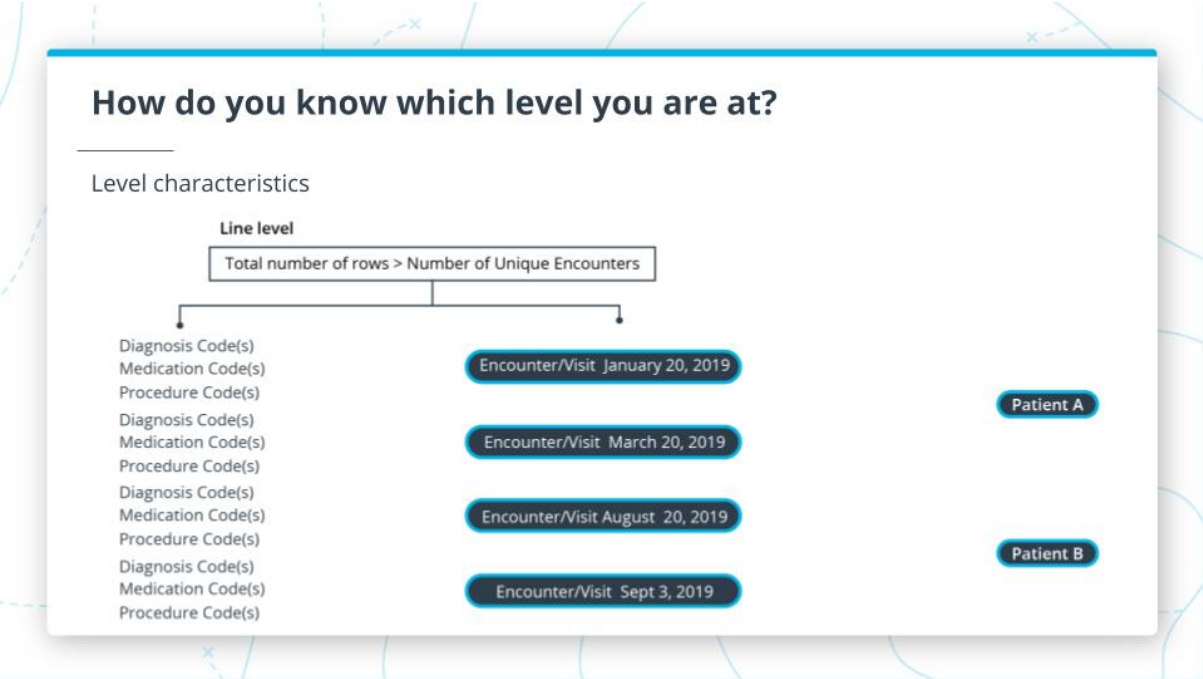
How do you know the dataset level for your data?

This is actually fairly easy if you collect some key metrics from your dataset and there are different ways to do this but I provided a few simple ways to do below.

1. The total number of rows in the dataset. This is a simple calculation with `len()`
2. The number of unique encounters or visits. You can calculate this by finding the field(s) that give the identity of a unique encounter using `nunique()`.

Example

```
total_rows = len(fake_df)
total_encounters = fake_df['encounter'].nunique()
```



Line Level

From here we do some simple calculations to figure out our dataset level.

If the total number of rows is greater than the number of unique encounters, it is at the line level.

Again using our example from above:

```
total_rows = len(fake_df)
total_unique_encounters = len(fake_df['encounter'].nunique())
```

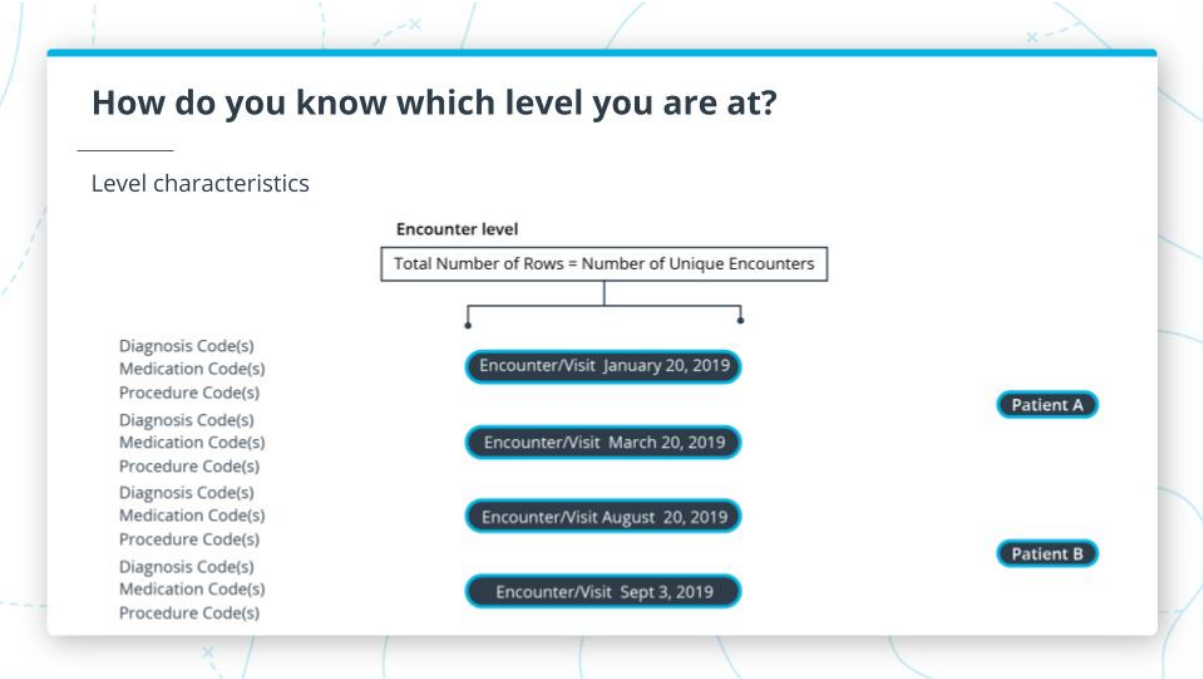
if the output was

- `total_rows = 43464`
- `total_unique_encounters = 3259`

We could find out using

- `print(total_rows > total_unique_encounters)` would evaluate to `True`

Therefore this dataset would be at the line level.



Encounter Level

If the total number of rows is equal to the number of unique encounters, it is at the encounter level.

Again using our example from above:

```
total_rows = len(fake_df)
total_unique_encounters = len(fake_df['encounter'].nunique())
```

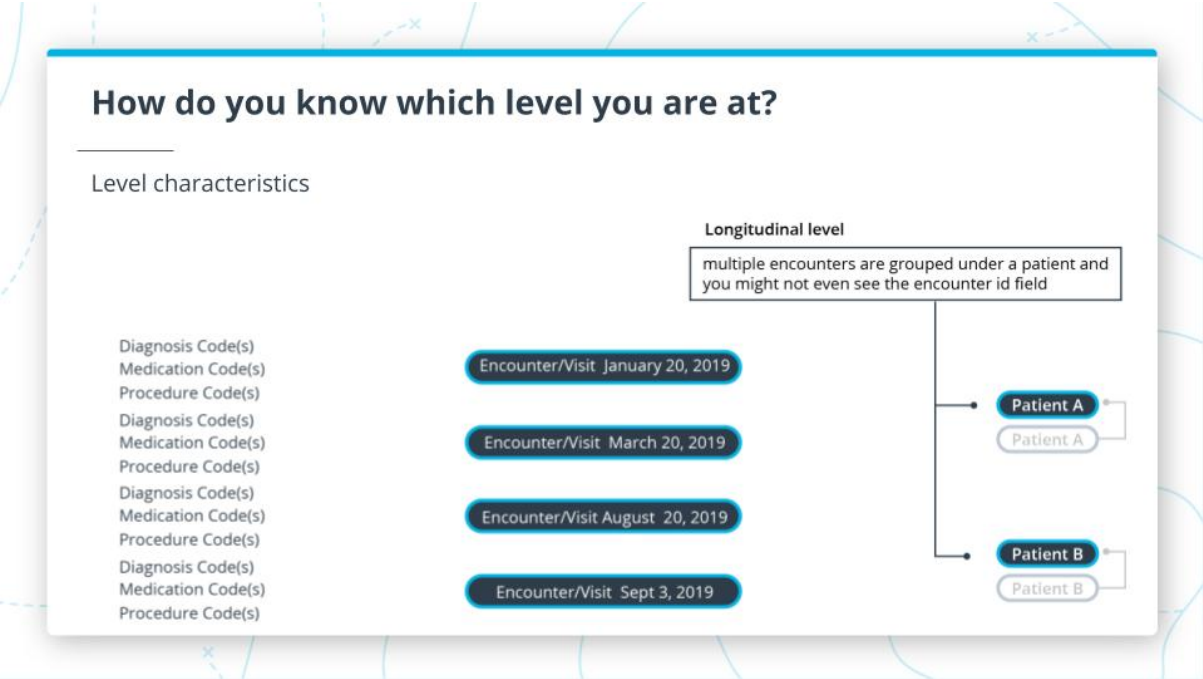
if the output was

- `total_rows = 3464`
- `total_unique_encounters = 3464`

We could find out using

- `print(total_rows == total_unique_encounters)` would evaluate to `True`

Therefore this dataset would be at the encounter level.

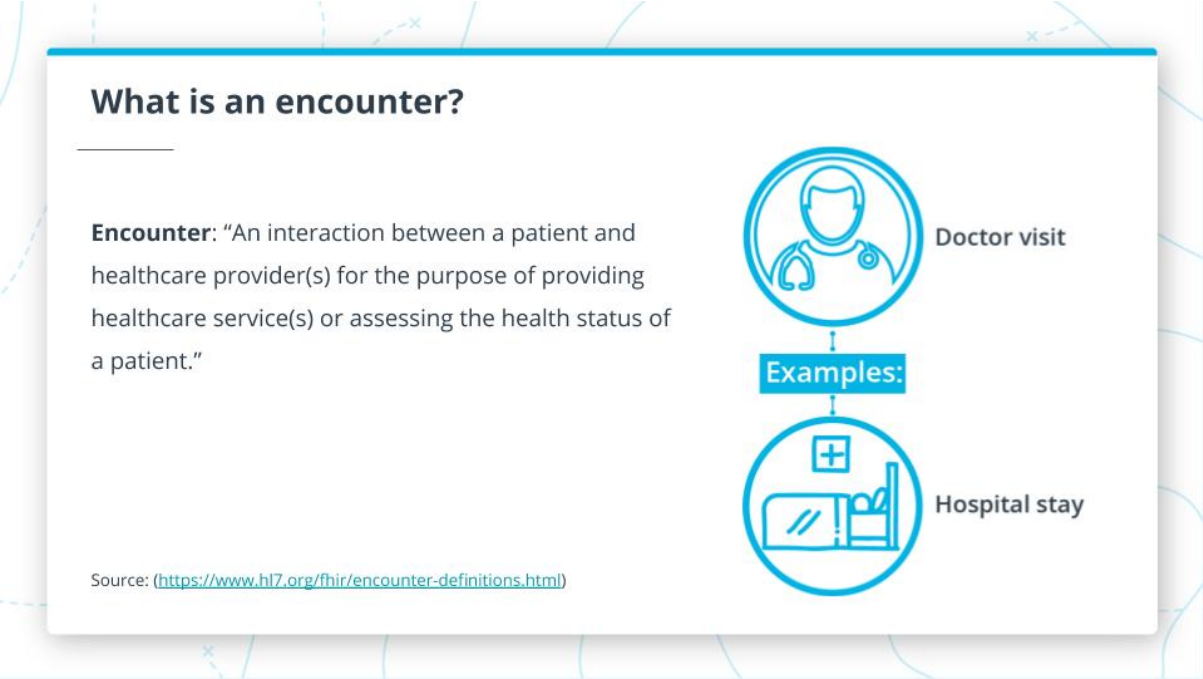


Longitudinal Level

Longitudinal Level

For the longitudinal or patient level, you will see multiple encounters grouped under a patient and you might not even see the encounter id field since this information is collapsed/aggregated under a unique patient id. In this case, the total number of rows should equal the total number of unique patients.

Encounter Representation



What is an Encounter? Encounter Definitions

Encounter: “An interaction between a patient and healthcare provider(s) for the purpose of providing healthcare service(s) or assessing the health status of a patient.”

What is an encounter?

The definition of an encounter commonly used for EHR records comes from the Health Level Seven International (HL7), the organization that sets the international standards for healthcare data. As the definition states, it is essentially an interaction between a patient and a healthcare professional(s). It usually refers to doctors visits and hospital stays.

How do we aggregate line level at encounter level?

1. Create a column list for the columns you would use to group. Likely these would be:
    - "encounter_id"
    - "patient_id"
    - "principal_diagnosis_code"
2. Create column list for the other columns not in the grouping
3. Transform your data into a new dataframe. You can use `groupby()` and `agg()` functions for this
4. Then you can do a quick inspection of the result by grabbing one of the patient records and to compare the output of the original dataframe and the newly transformed encounter dataframe.

Note: The data used in the walkthrough was created just to show you how this would work. You'll practice this more in later exercises as well.

Additional Resources

- **Encounter Definitions**

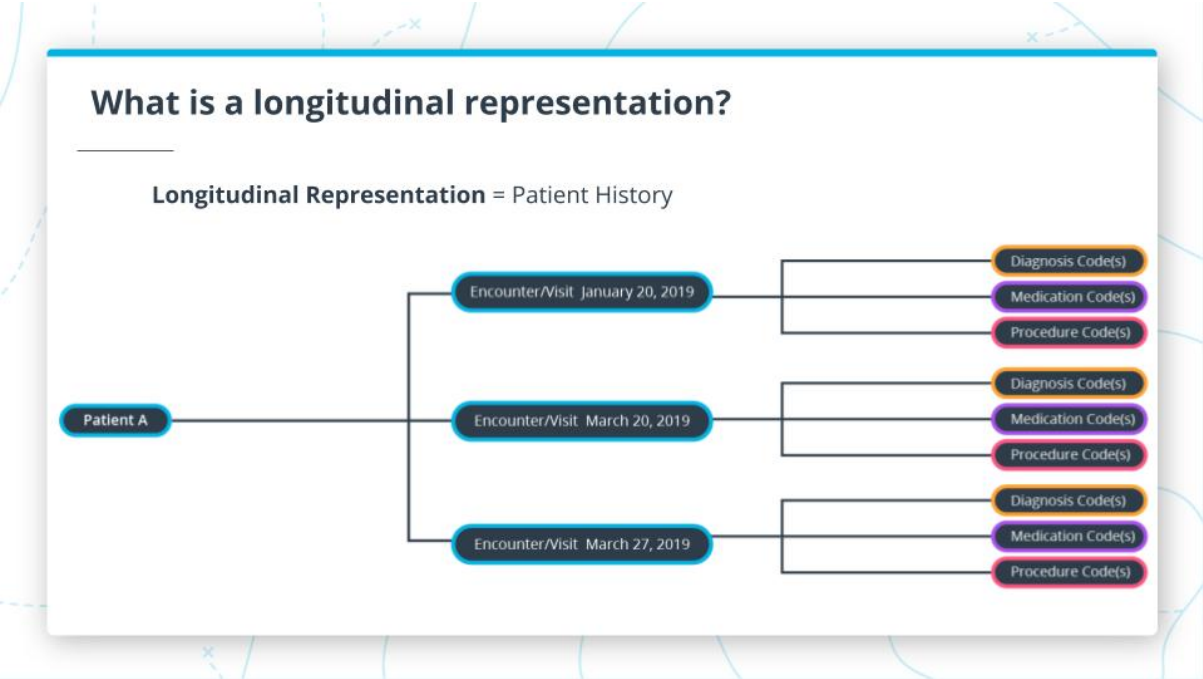
Longitudinal Representation

Longitudinal Representation: Patient History

What is a longitudinal data representation?

Another way to view it is a patient history representation. It is basically a way to aggregate all of the visits/encounters that a patient may have in the healthcare system. Having all of this information is ideal to best analyze and diagnose correctly, but the reality is that patient records do not always come cleanly organized and aggregated correctly, in real-world datasets. Luckily, we can use the line and encounter levels to help with that!

Longitudinal Representation Example



Combining patients together into a dataset and finding a way to aggregate and represent the visits across a patient is the key to being able to unlock powerful insights and analysis.

Challenges and Benefits of Using the Longitudinal Level

Challenges:

- Scaling with this type of data requires vast resources
- Data representation and preparation requires more preparation and validation
- Training vs production/deployment data differences can cause major issues with model predictions

Example: You train this great Model, and it has fantastic results!

However, the data you used for this model had an implicit assumption that you were unaware of :

Only the label for the last encounter for a patient history would be included.

You think that you can always transform the data in production. Right?

Reality: You have put your model into production and things start going haywire.

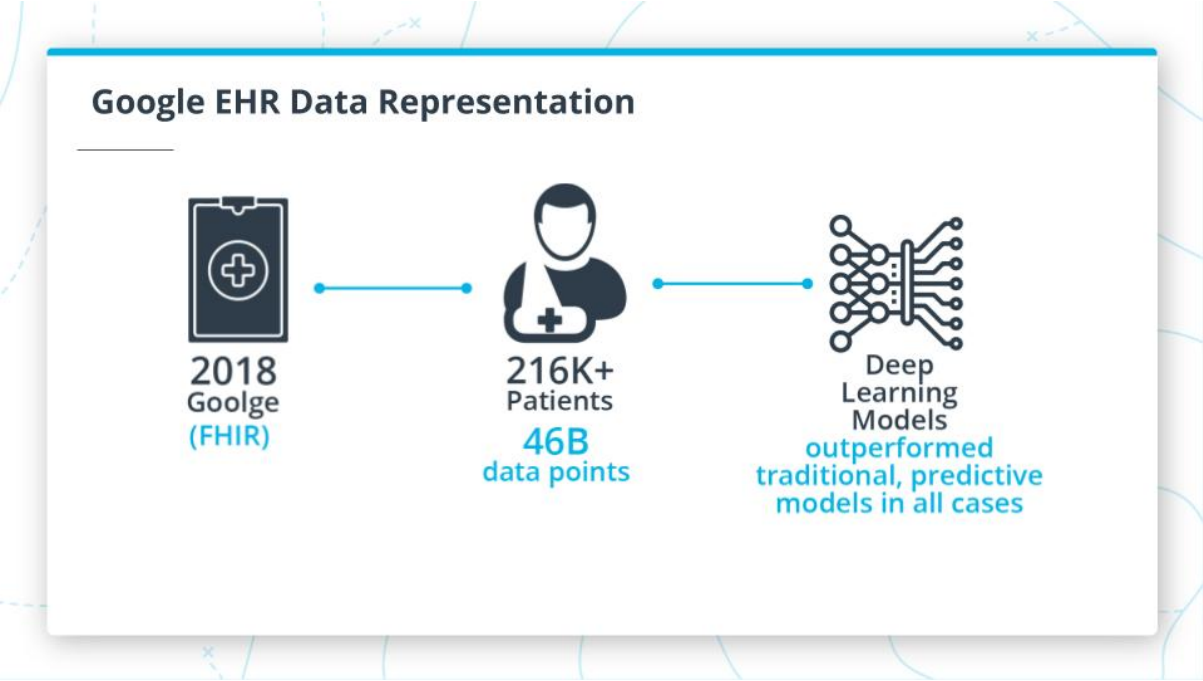
You are not getting the same results that you anticipated from your trained model. Even with some cross-validation and other augmentation methods, you still see significant model drift. After some analysis of the production data pipeline, you find out that the production use case has an **Encounter selected at random** from the patient history and information has been duplicated by accident because the information was not leveled correctly.

From this example, you can see how important it is to make sure you, or your data team, have properly aggregated the data.

Benefits:

Even though data preparation and validation are more difficult, there are significant benefits to using a longitudinal representation as it allows you to use a full patient history. This helps to better model changes in state over time. When you can also aggregate other patients longitudinal data together, you can achieve some unbelievable results.

Deep Learning Potential Importance



To further illustrate the importance of having a strong EHR longitudinal data representation, you can look through the links in this section about Google creating a data representation to feed into deep learning models from the Fast Healthcare Interoperability Resources or **FHIR** standards.

In Google's ***Nature*** article, they highlighted the aforementioned challenges with building a longitudinal representation. The 216K+ patients in the dataset were extrapolated to 46B data points! Now imagine attempting to scale this even further with more patients or other types of data like genomic data, clinical trial data, etc. This is where resources like cloud computing have become critical, but even these can be strained with this much data.

As most practitioners know, the data preparation stage is one of the most time consuming, but impactful parts of a data science project. The representation and deep learning models outperformed clinically used traditional predictive models in **ALL** cases. It is still early days for the healthcare industry as they have just started to consider using deep learning and still barriers exist in using them due to concerns about interpretability. Hopefully, with more studies like this, critical attention to detail and eye on data security and privacy we can bring this practice to the forefront in healthcare.

Additional Links

- **Google's Nature Paper**
- **Google's Patent**

- **FHIR**
- **FHIR Overview**

How to Convert to Longitudinal Level

Note: In this walkthrough, we continued to use the synthetic dataset from the previous walkthrough.

1. Inspect the data!

Remember that when working with longitudinal data it is vitally important that we inspect the data.

1. Create a new dataframe that groups the data by "patient_id". Again you can use `groupby()` and `agg()` methods.
2. Inspect and compare the data again by selecting a single patient and compare the "encounter", "principal_diagnosis", and "procedure_codes".

You should see all of these codes represented in arrays/lists for each patient.

Example:

```
list(longitudinal_example_patient_history['encounter_id'].values)
```

```
Output: [['encounter_id_2352', 'encounter_id_7434','encounter_id_1393']]
```

Dataset Splitting Without Data Leakage

This may seem like a trivial topic, however, it is actually something that is often done incorrectly and can lead to significant downstream issues later.

Imagine that you spent significant time working on a model only to find out that your results are invalid because the data was not prepared correctly. This sometimes happens due to a rush to process data without enough testing and validation of the data and steps before modeling.

Challenges:

Data Leakage: Inadvertently sharing data between test and training datasets.

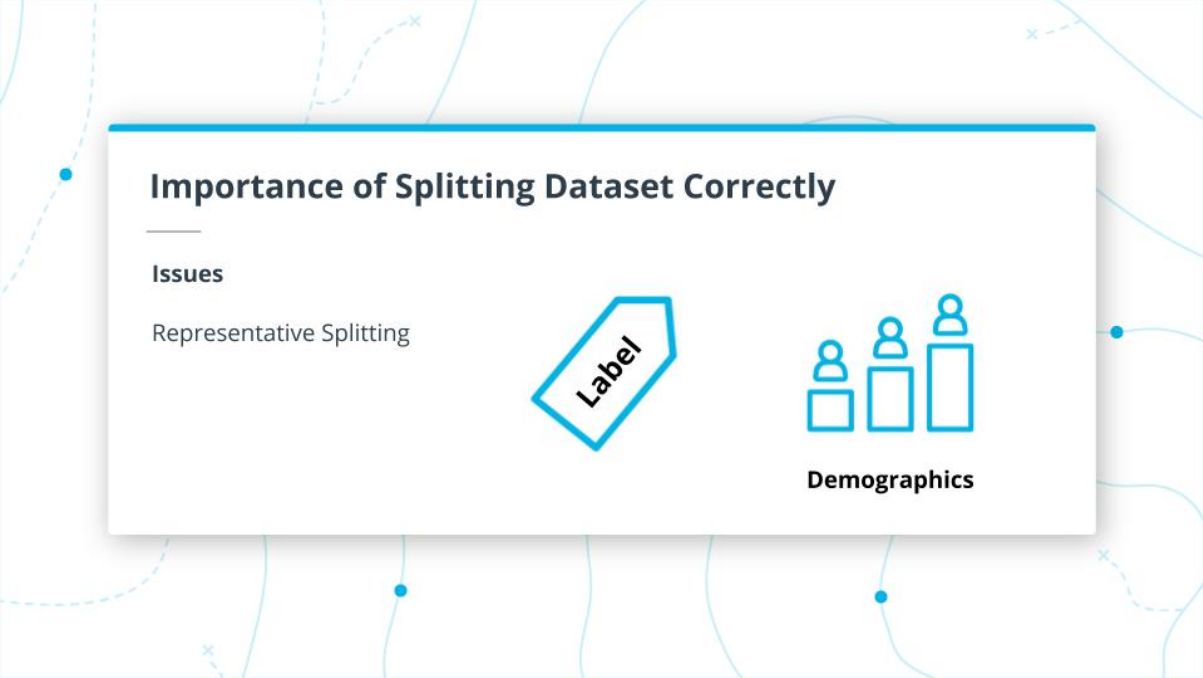
Data leakage is a massive problem as your model will perform fantastically during training and fail miserably in production.

An example of where this can occur is when you have a longitudinal dataset and you use different patient encounters across different splits. You may inadvertently leak in information about the patient into your training data that you will be testing on. Essentially giving your model some of the answers. So preventing data leakage is very important to ensure your mode can generalize in production.

Additional Resources

- Kaggle Preventing Data Leakage

Representative Splitting



Challenges continued

Representative Splitting: Having accurate labels and demographics in your data splits that reflect the real world. A major challenge for most machine learning problems is a generalization and building a dataset that is representative. Common errors that can occur include:

- Non-representative distribution of your label across the splits
- Non-representative demographics

Example: Only female patients in training and male in testing

This would introduce some unintended biases and issues in your model for procedures that are gender-specific.

Testing and Validating Dataset Splitting

It is important to have some ways to assess whether you have split your data right.

Here are a few ways to do this.

1. Assess to make sure that a single patient's data is not in more than one partition to avoid possible data leakage.
2. Check that the total number of unique patients **across the splits** is equal to the total number of unique patients in the **original dataset**. This ensures no patient information lost in the splitting and that the counts are correct.

3. Check that the total number of rows in **original dataset** should be equal to the sum of rows across **all three dataset partitions**.

`len(original_df) == len(train_df) + len(val_df) + len(test_df)` should evaluate to `True`.

Transform/Preprocess Numerical Features with Feature Column API

The TensorFlow Feature Column API helps make data preprocessing easier by abstracting away some of the work for things like normalization in numerical features. If you have done this type of work in Scikit Learn or Pyspark, you might appreciate the work this API does for you when it comes to preparing features for modeling. It also has the ability to add less common features like cross features and shared embeddings.

Additional Resources

- **TensorFlow Feature Column API**
- **Learn more about Cross Features**
- **TensorFlow Cross Features**
- **TensorFlow Shared Emeddings**

Categorical Features & Feature Column API

For building categorical features with the TensorFlow Features Column API, we follow the following steps:

1. Select the categorical feature columns.
2. Create a vocabulary text file with all of the unique values for a given categorical feature and add a placeholder value for out of vocabulary(OOV) values in the first row.
    - **Note**: For columns with a small number of features, you probably don't need to do this step and can instead pass an array/list to the **categorical_column_with_vocabulary_list function**.
    - The creation of a separate vocabulary file method is particularly great for features with high cardinality. It can also allow you to use other tools like SQL to generate these vocab files with more massive datasets and decouple this process if you are already creating these for data profiling purposes.
3. Create your new feature by passing the vocabulary feature to the final derived feature. This can be an embedding or one-hot encoded feature. In this example, we created a one-hot encoding feature with the **indicator column function**.

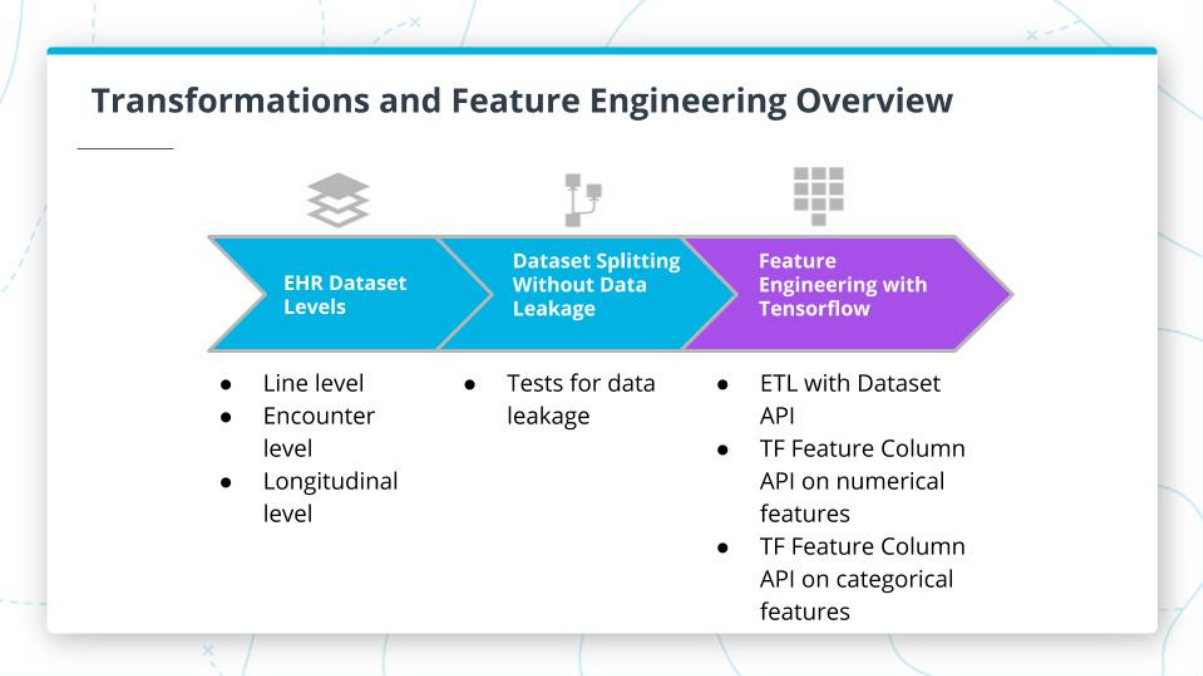
Example:

```
principal_diagnosis_vocab = tf.feature_column.categorical_column_with_vocabulary_file(
            key="PRINCIPAL_DIAGNOSIS_CODE", vocabulary_file = vocab_files_list[0], num_oov_buckets=1)
```

Additional Resources

- **Tensorflow Feature Column API Categorical Features**
- **TensorFlow One-hot Encoding**

Transformations and Feature Engineering Recap



Fantastic Job! You are 75% of the way through this course!

In this lesson you learned:

- to identify and conduct tests to determine the correct EHR dataset level. You learned to differentiate between the line, encounter, and longitudinal levels. This skill will ensure that as we build our models, you are using the right data level.
- conducted some basic tests while splitting data like a lumberjack preventing Data leakage which can be a major issue in EHR datasets.
- engineered features with Tensorflow. In this part, you performed ETL with the Dataset API. You also transformed datasets using the TF Feature Column API for both numerical and categorical features.

You can now use the Feature Column API for transforming datasets at scale and building some unique feature types.

Amazing job! The next lesson will be even more fun as we start building our machine learning models!

Lesson Key Terms

Aa Key Term	Definition
Line Level	A denormalized representation of all the things that might happen in a medical visit or encounter.
Encounter Level	Also known as the visit level, which is the aggregated information from the previously mentioned line level for one encounter. This information can be collapsed into a single row or arrays.
Longitudinal Level:	Also known as the patient level view. This level aggregates the patient history and can show how the culmination of visits/encounter lead to some clinical impact.
Encounter	“An interaction between a patient and healthcare provider(s) for the purpose of providing healthcare service(s) or assessing the health status of a patient.”

Building, Evaluating and Interpreting Models

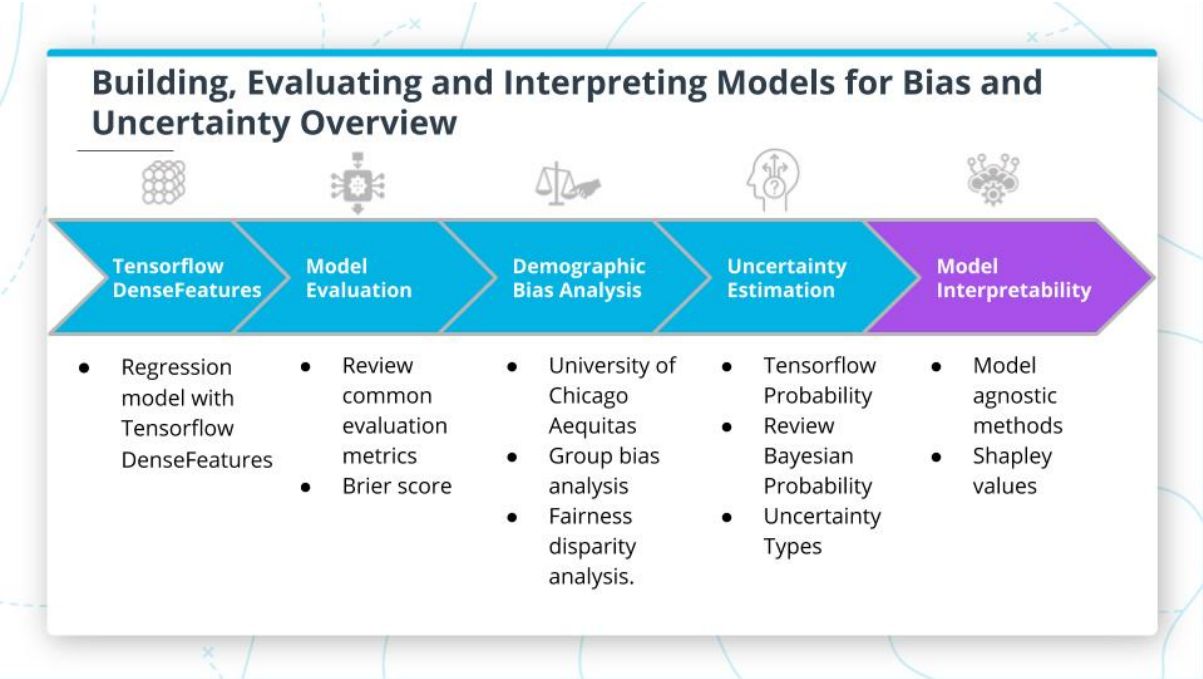
Building, Evaluating & Interpreting Models Overview

This lesson contains some really fantastic tools for working with EHR data in AI.

Note: At the time this material was created some of the TensorFlow features are in beta form and may change as they launch. That being said these are some bleeding-edge resources to help with many AI projects.

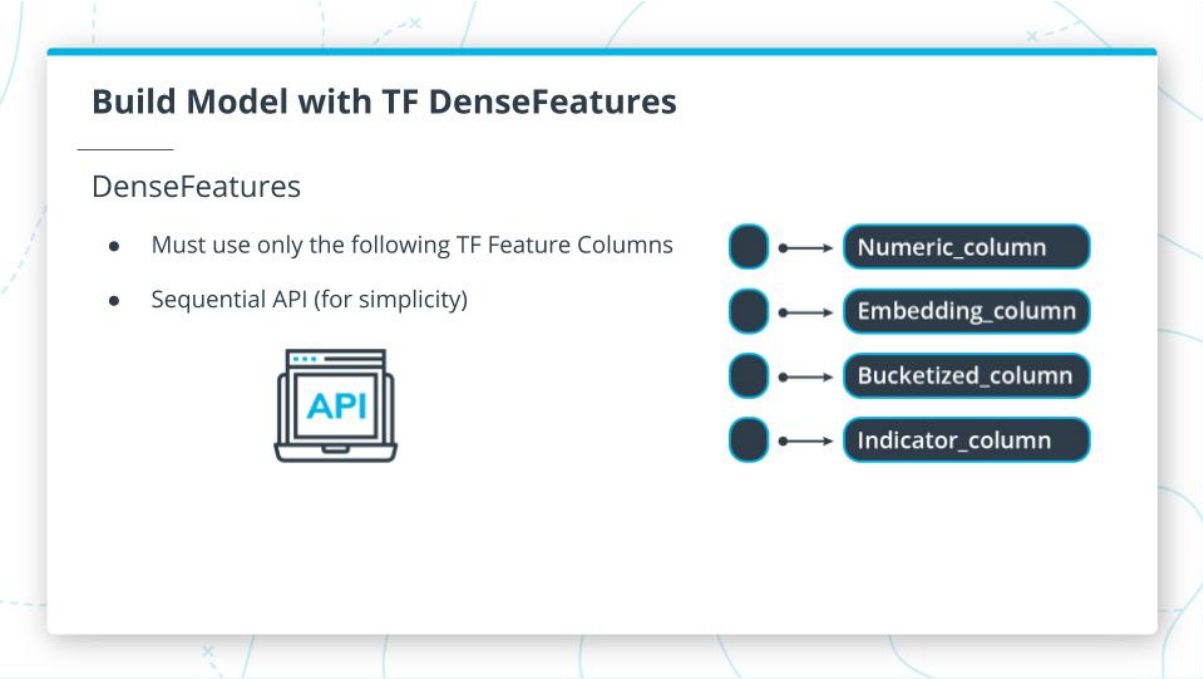
- In the first part, you will get hands-on with using Tensorflow `DenseFeatures` for building a simple regression model.
- Next, you will first review some common evaluation metrics for EHR models and then learn to implement brier scores for model evaluation
- Then, you'll conduct a demographic bias analysis and become familiar with a framework out of the University of Chicago called Aequitas. You will use this Aequitas for group bias and fairness disparity analysis.
- Next, you will implement uncertainty estimation using the Tensorflow Probability API. You will also review some of the underlying concepts for Bayesian probability and types of uncertainty that very important in evaluating model performance.
    - **Note**: TensorFlow Probability is still in beta at the time this course was published. Documentation is still being built out and some of the code may change.
- Finally, you will interpret models with Shapley values. You will first review model interpretability and some model agnostic methods. Then you will use one of those methods, Shapley values.
    - **Note**: This section is included as a bonus and is not included in the project itself, but is highly useful.

There is so much to cover in this section. Head to the next section to get started.



Lesson Overview

Tensorflow Regression Model with DenseFeatures



Build Model with TF DenseFeatures

TF DenseFeatures Key Points

In this section, we will focus on just building a simple Tensorflow regression model with TF `DenseFeatures`, but we realize that there are more advanced deep learning architectures. Please feel free to try different approaches to augment your features and architecture with the walkthroughs and exercises in this lesson. There are some additional links below to explore

around with some AutoML offerings. These are not the focus of this course but are included in case this is interesting to you.

Build Model with TF DenseFeatures

Tensorflow DenseFeatures, combining features, like those from the TensorFlow Feature Columns API, into a dense representation for the model.

You can only use certain TF Feature Columns with `DenseFeatures`:

- `numeric_column`
- `embedding_column`
- `bucketized_column`
- `indicator_column`

Note:

- For the sake of simplicity, we will use the Sequential API for this course, but if you want to customize further, feel free to try to the Functional API. However, you might encounter some issues later with configuring some other parts with the TF Probability outputs.
- As of writing, Tensorflow is experimenting with Sequence Features columns as well that can be combined with the `SequenceFeatures` function. This will not be covered in the course but wanted to share this for your info since it can be very useful.

Additional Resources

- TF DenseFeatures
- Functional API
- SequenceFeatures
- GCP AutoML
- AWS Autopilot
- AdaNet
- AutoKeras
- H20 Driverless AI
- AdaNet Additional
- AutoKeras Titanic
- Evolved Transformer

TF DenseFeatures Walkthrough

Tensorflow DenseFeatures Layer

Tensorflow provides the **DenseFeatures layer** as a way to combine some of the Tensorflow feature columns we learned earlier into a dense input tensor to your model. For this example, we use the **Sequential API** but you could use the **Functional API** as well with some

adjustments. Hopefully, the Tensorflow Keras modeling functions should be familiar to you and for the walkthrough, we again used the architecture from the **Tensorflow regression tutorial**.

One key thing to note is that you can only use some Tensorflow Feature Column types with DenseFeatures so it is worth checking. At the time of writing this course, here are the features that you can use.

- **Numeric Feature**
- **Embedding Feature**
- **Bucketized Feature**
- **One hot encoded Feature**

Below I will walk through the simple steps to use the DenseFeatures layer.

- First, combine all of your Tensorflow feature columns into a list that can be passed to the DenseFeatures layer. Please note that this is not the column name but the Tensorflow feature column metadata that you will get when you create a Tensorflow feature.

```
feature_columns = tf_categorical_feature_list + tf_numerical_feature_list
dense_feature_layer = tf.keras.layers.DenseFeatures(feature_columns)
```

- Next, we simply add this 'dense_feature_layer' as the first layer to the model and this will handle the combining of feature inputs to the model.

```
  model = tf.keras.Sequential([
  **dense_feature_layer**,
  tf.keras.layers.Dense(64, activation='relu')
 ... additional layers
```

Common EHR Model Evaluation Metrics

For this course, we will assume some exposure to common evaluation metrics used for classification and regression. These terms should hopefully be familiar and we will not cover them fully, but here is a quick review.

Common Classification Metrics

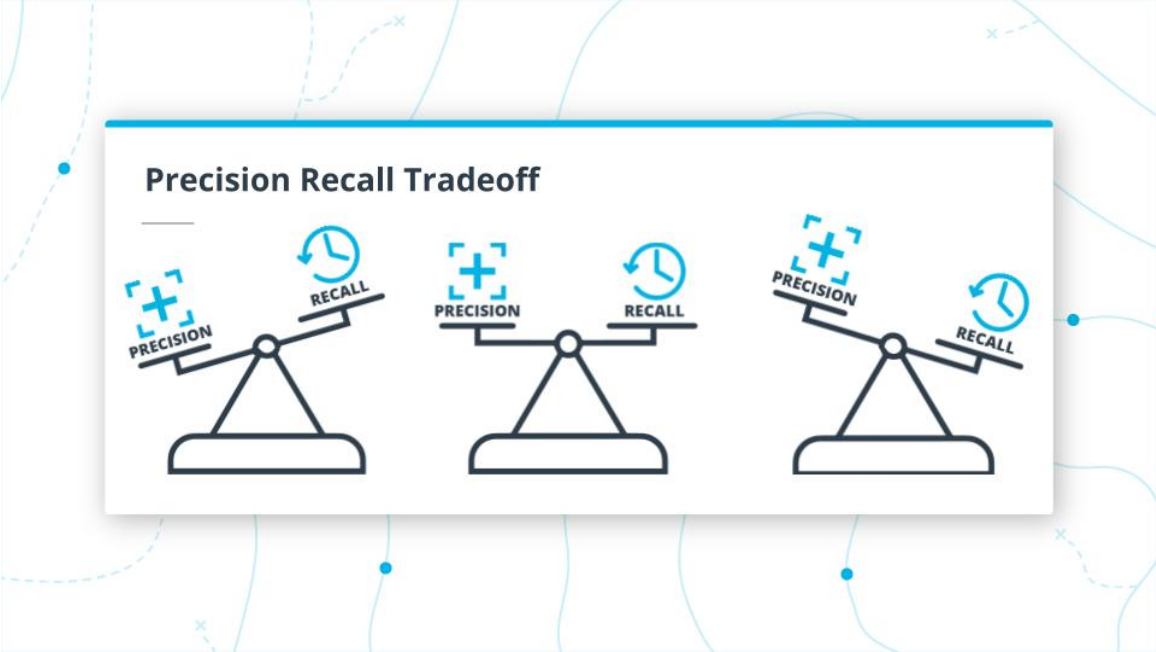
- **ROC**: receiver operating characteristic curve or ROC curve that shows a graph of the performance of a classification model. It is the True Positive Rate Vs. False Positive Rate across different thresholds.
- **AUC**: Area under the ROC Curve measures the entire two-dimensional area underneath the entire ROC curve.
- **F1**: Harmonic mean between precision and recall
- **Precision**: the fraction of relevant instances among the retrieved instances

- **Recall**: the fraction of the total amount of relevant instances that were actually retrieved.

Common Regression Metrics

- **RMSE**: Root Mean Square Error- a measure of the differences between values predicted by a model.
- **MAPE**: Mean Absolute Percentage Error is a measure of quality for regression models loss.
- **MAE**: Mean Absolute Error is a measure of errors between paired observations.

Precision-Recall Tradeoff



Precision Recall Tradeoff

As a quick reminder, there is often some level of precision that you must align with and some capture rate or recall that you are trying to improve. This balance between the two is a necessary tradeoff and a key component that you must communicate with non-technical stakeholders who might have unrealistic views of the impact or capabilities of AI.

Brier Scores

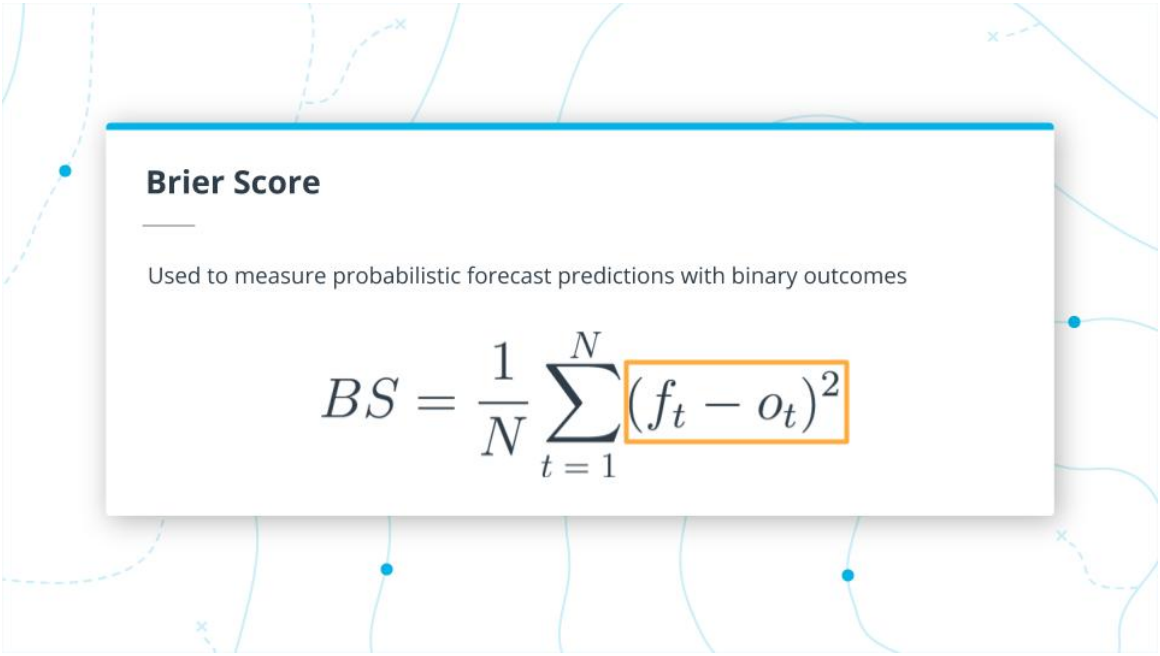
One metric that you might not be familiar with is a Brier score and it is often used in weather forecasting for estimating the probability certainty of a forecast. It can be a useful metric for comparing the performance of algorithms based on the degree of confidence in a given prediction. This can be helpful because the confidence and measurement of uncertainty can yield vastly different interpretations.

For the actual definition, it is essentially the mean squared error of a given probability forecast and I walk through the formula step by step below. However, please note that you are not expected to know this formula or use it in this course because we are focusing on

the predictions for a regression model. It is introduced as a relevant evaluation metric for if we had a binary classification problem with the associated prediction probabilities.

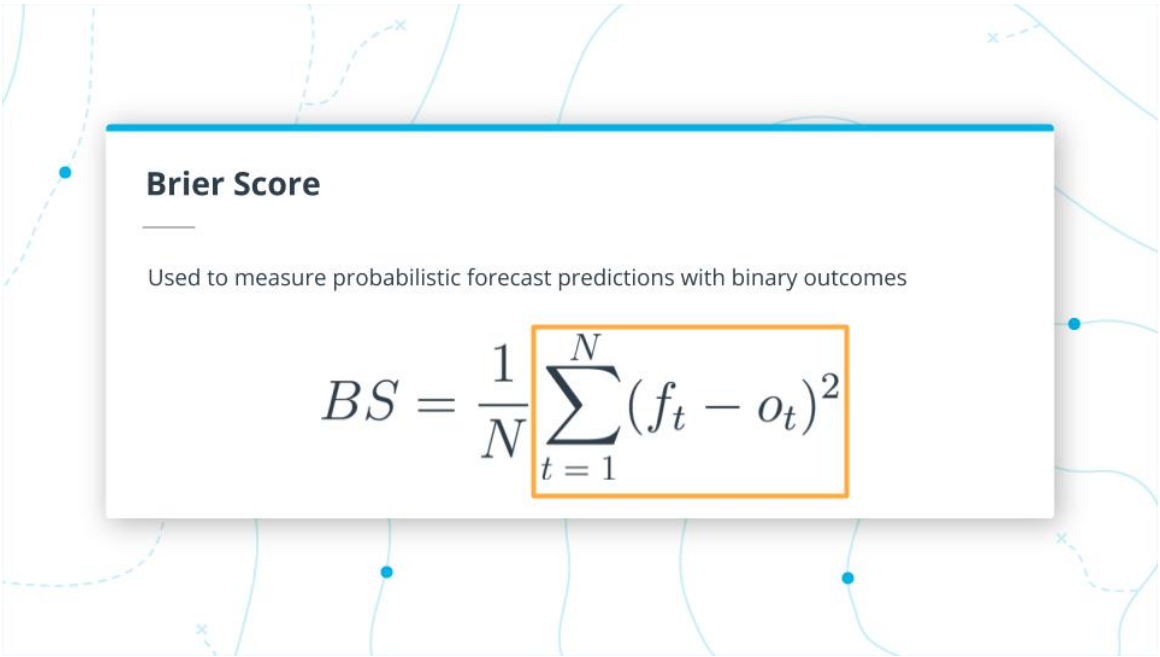
Brier Score Breakdown Part 1

Basically you take the probability forecast on a 0 to 1 scale which is *f of t* and subtract that from *O of t* which is the actual value which is a binary 0 and 1 value. You then square the difference of this value.



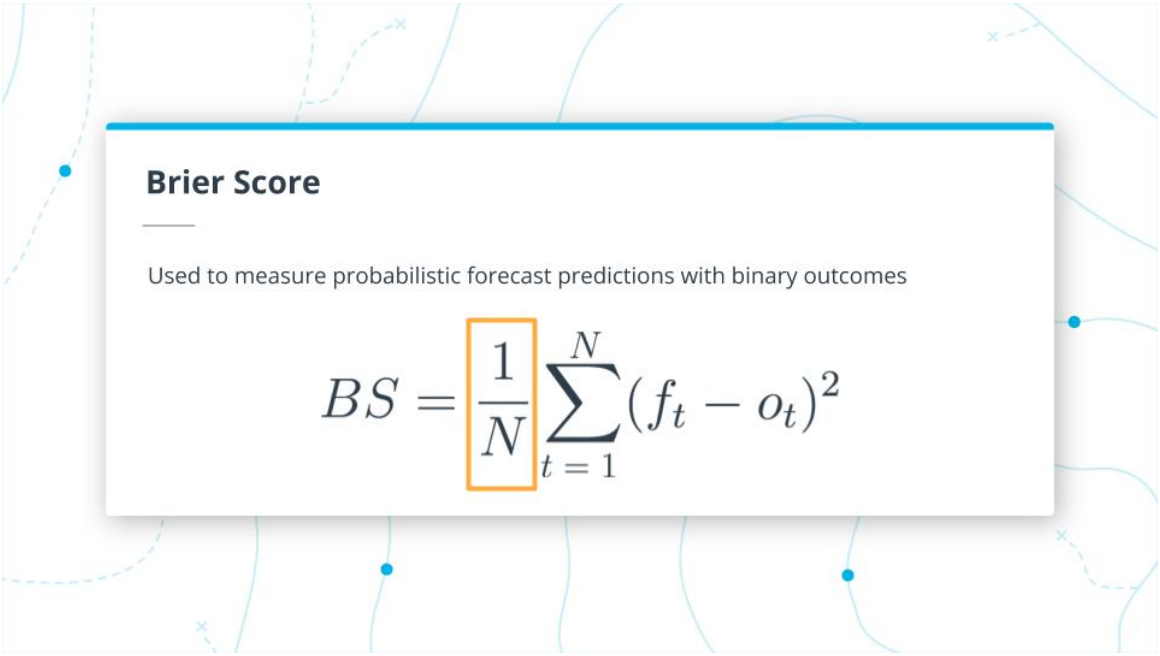
Brier Score Breakdown Part 2

Take the summation of these squared differences from t=1 to N, the total number of predictions.



Brier Score Breakdown Part 3

Then divide by N the total number of predictions



Brier Score Breakdown Part 4

The result is the Brier Score which ranges from 0 and 1.

- Lower is better
- 0 is the best score
- 1 is the worst score.

Additional Resources

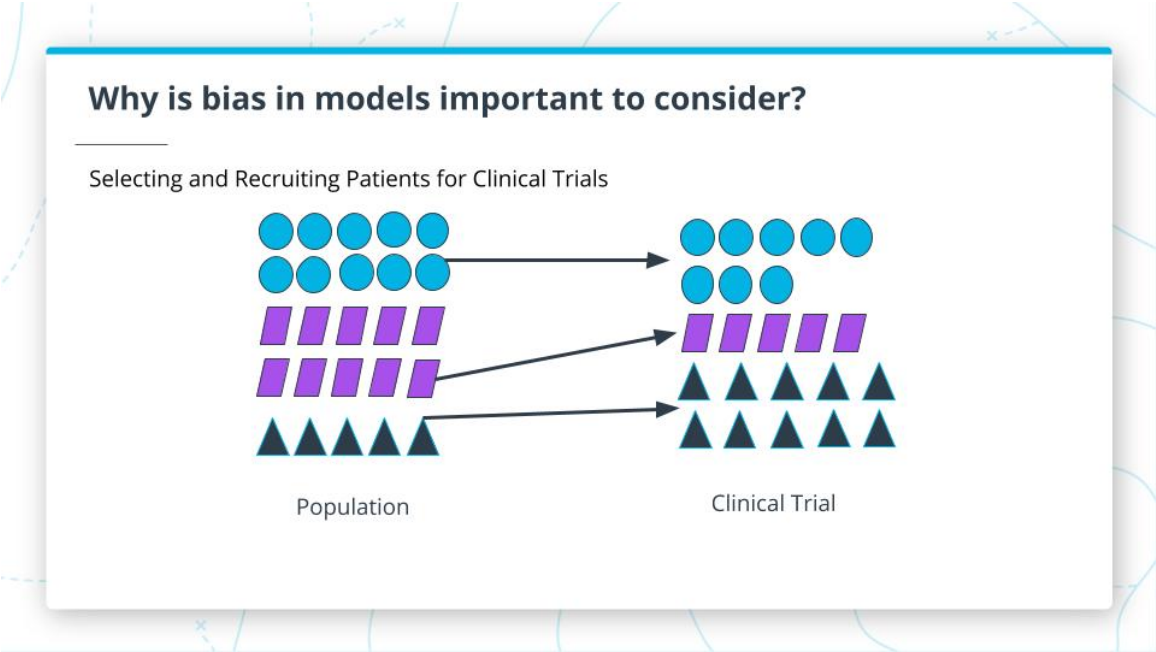
- **A Note on the Evaluation of Novel Biomarkers: Do Not Rely on Integrated Discrimination Improvement and Net Reclassification Index**
- **Brier Scores**
- **Brier Score Limitations**

Demographic Bias Analysis

Why is bias important to consider in your models?

It's important to note that bias within models can restrict or limit patient access to key medical benefits from government aid programs.

Programs using AI algorithms to help automate approvals of key government benefits are becoming more commonplace. However, it is just as important to consider how bias can unintentionally occur.



Why is bias in models important to consider?

Another reason that you want to consider bias in models is that in order to create better treatments for patients we need to find better ways to select and recruit patients that represent the wider population that a drug/treatment would be targeted for.

In many cases, there may be systemic biases for key groups and while this cannot always be prevented, bringing awareness of limitations and biases can give a more accurate picture of a treatment's effectiveness across different demographics.

In the example above, we can see that the purple parallelogram is representative of the population, while the blue circles and triangles are not and both are over-represented in the trial.

Unintended Bias

Unintended biases: a bias that is not intentional and often is not even apparent to the creator of a model

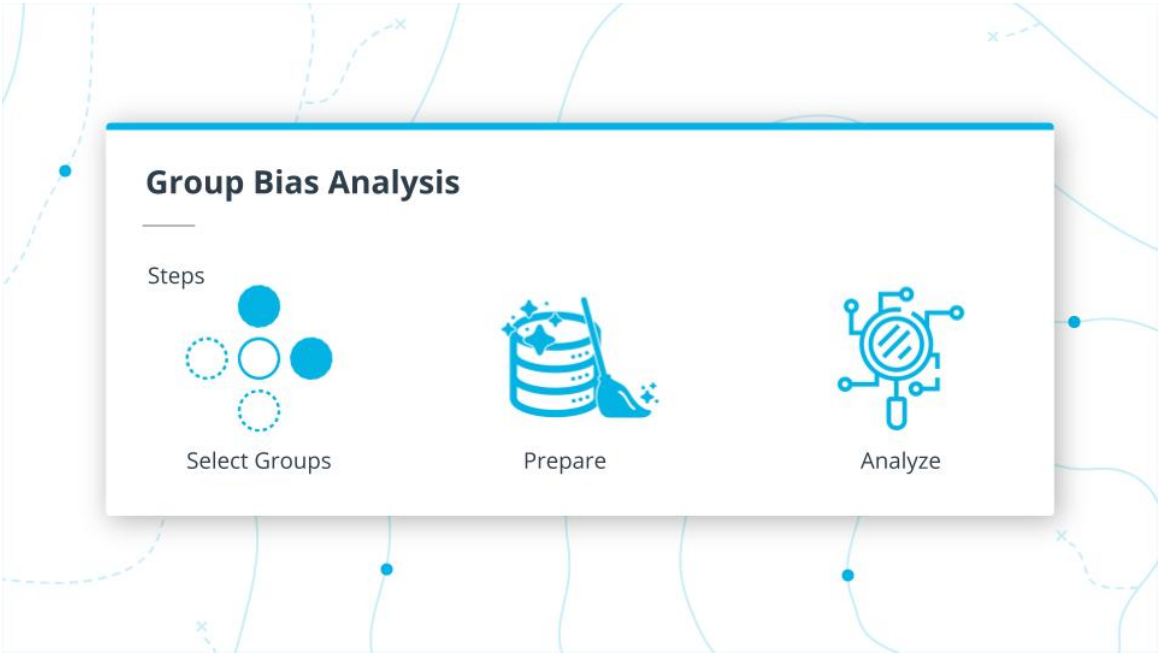
Unintended biases represent the unconscious or unintentional biases that come with the AI models that we are building. Becoming more aware of these biases and how they impact different groups is key.

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.

Aequitas

- Developed by University of Chicago Data Science for Social Good
- Addresses concerns about unintended bias unfairly affecting certain groups
- Definitions and metrics for unintended bias in predictive models

Demographic Bias Analysis Walkthrough



Group Bias Analysis

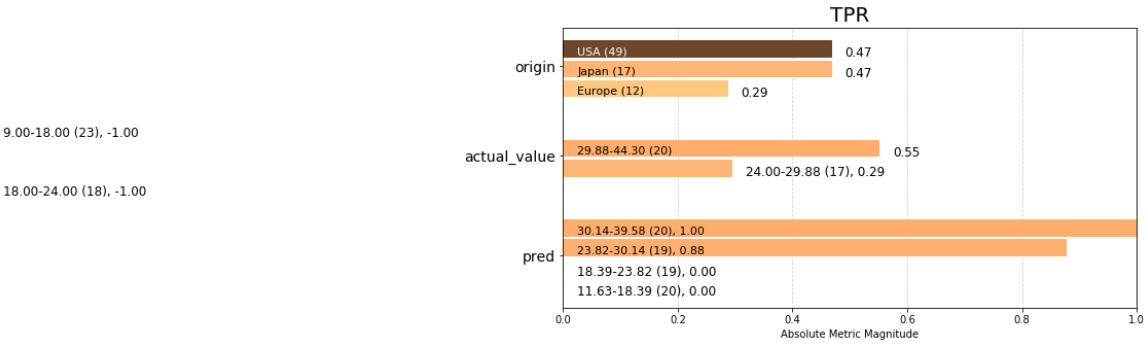
Steps for Demographic Group Bias Analysis

- **Select the groups we want to analyze**: For this walkthrough, we selected the 'origin' field as a group to analyze and in this case it is for the country of origin for a car.
- **Prepare the data:** We use the boilerplate preprocessing code provided that takes a Pandas dataframe with the 'score' and 'label_value' fields.

```
from aequitas.preprocessing import preprocess_input_df
from aequitas.group import Groupfrom aequitas.plotting import Plot
from aequitas.bias import Bias
from aequitas.fairness import Fairness
ae_df, _ = preprocess_input_df(merged_binary_df)
g = Group()
xtab, _ = g.get_crosstabs(ae_df)
absolute_metrics = g.list_absolute_metrics(xtab)
clean_xtab = xtab.fillna(-1)
aqp = Plot()
b = Bias()
```

- **Analyze different metrics with the groups**: Then we can do different analyses such as TPR with the following function.

```
tpr = aqp.plot_group_metric(clean_xtab, 'tpr', min_group_size=0.05)
```

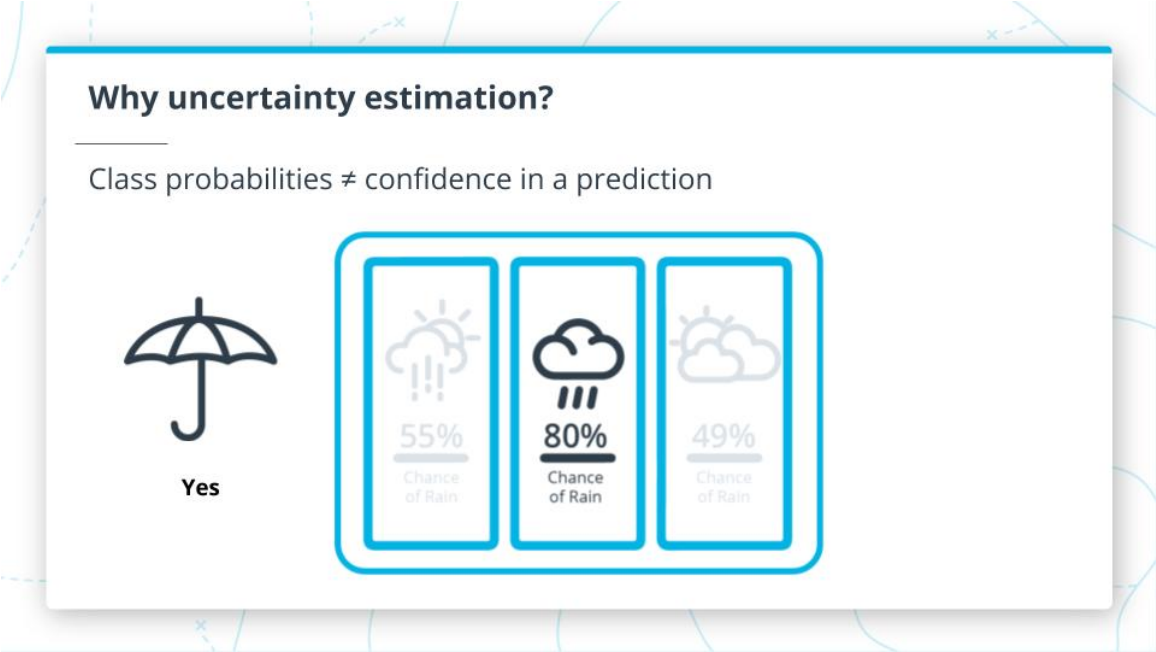


Uncertainty Estimation

Why use Uncertainty Estimation?

A typical classification problem provides predictions with relative weightings across the prediction classes and is different than the confidence in a given prediction. Uncertainty estimation helps us with this.

Class Probabilities != confidence in a prediction.



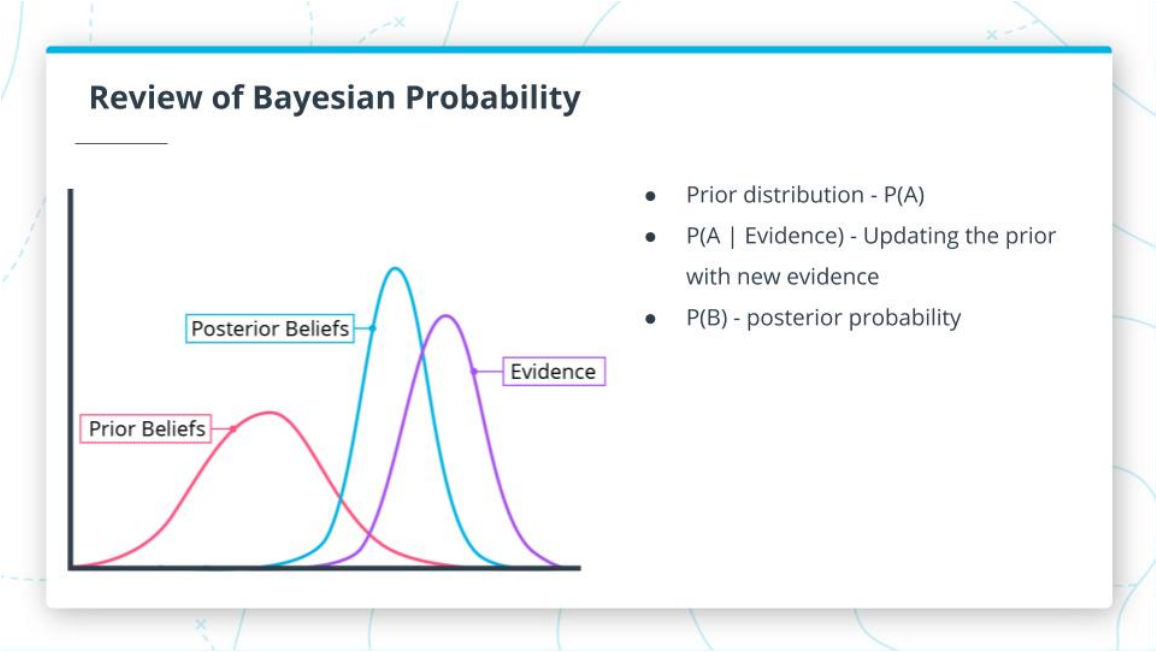
Uncertainty Weather Example

Example:

Which is better?

1. A weather forecast with a simple Yes or No prediction it will rain with some icons?
2. A forecast that also provides the percent chance of rain or how certain the model is of its prediction?

Most people would prefer the second one and in healthcare this even more important.



Review of Bayesian Probability

Bayesian Probability Review

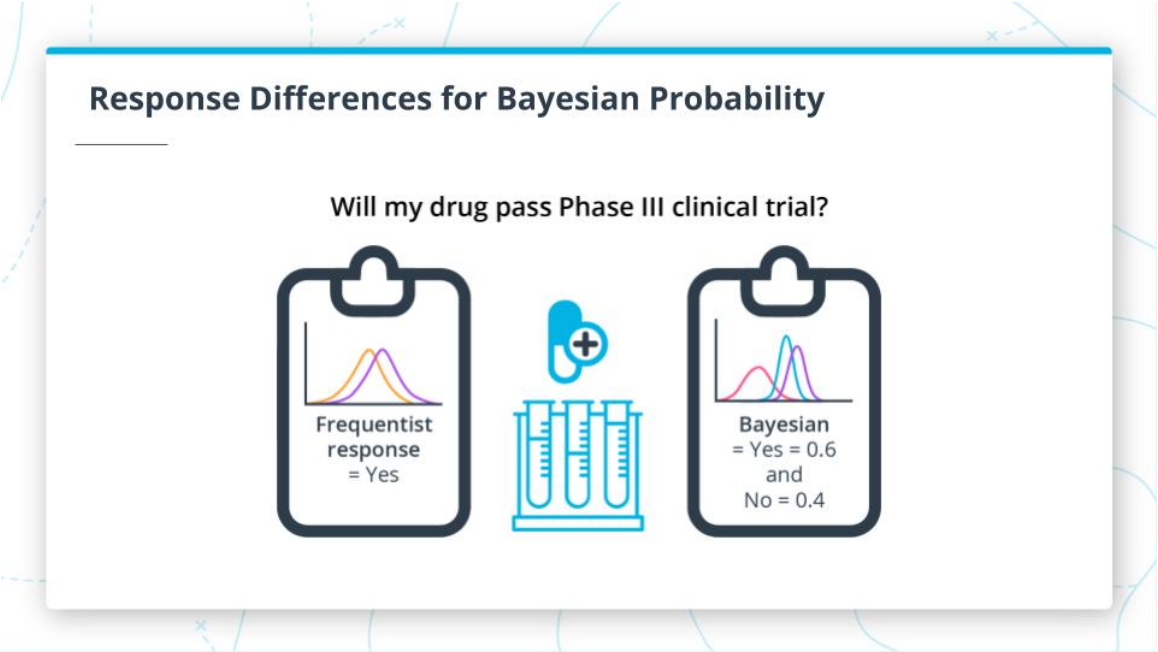
Probabilistic Programming

To address uncertainty estimation we can use probabilistic programming, in particular, we can create Bayesian Neural Networks (BNN) using TensorFlow Probability. TF probability combines Bayesian probabilistic approaches with deep learning and since it is built on Tensorflow you can train with GPUs too.

Bayesian statistical approaches can be very helpful for situations where you can leverage deep domain knowledge and have small datasets. Healthcare is an industry where you might have small datasets on patient data for a new drug or rare conditions. Also, healthcare is a field where you can leverage deep domain knowledge from medical professionals and medical literature which can help provide positive “bias” in your model with pertinent domain knowledge.

Review of Bayesian Probability

- Prior distribution - P(A)
- P(A | Evidence) - Updating the prior with new evidence
- P(B) - posterior probability



Response Differences for Bayesian Probability

Example:

Let's say we want to assess whether a drug will pass a phase III clinical trial. The frequentist statistical response would be a yes or no response. However, the bayesian response would give a yes/no and the certainty/confidence in that prediction. This can be helpful when connecting it with metrics like Brier scores that can combine predictions from various models.

If you are only 51% certain that you will pass the clinical trial, how certain would you be?

Types of Uncertainty

- Aleatoric:
  - Statistical Uncertainty - a natural, random process
  - Known Unknowns

Aleatoric uncertainty is otherwise known as statistical uncertainty and is known unknowns. This type of uncertainty is inherent, and just a part of the stochasticity that naturally exists. An example is rolling a dice, which will have an element of randomness always to it.

- Epistemic:
  - Systematic Uncertainty - lack of measurement, knowledge
  - Unknown Unknowns

Epistemic uncertainty is also known as systemic uncertainty and is unknown unknowns. This type of uncertainty can be improved by adding parameters/features that might measure something in more detail or provide more knowledge.

Additional Resources

- **Uncertainty**

Key Points

Building a Basic Uncertainty Estimation Model with Tensorflow Probability

- Using the MPG model from earlier, create uncertainty estimation model with TF Probability.
- In particular, we will focus on building a model that accounts for Aleatoric Uncertainty.

NOTE: Before we go into the walkthrough I want to note that TF Probability is not a v1 yet and documentation and standard patterns are evolving. That being said I wanted to expose you to a tool that might be good to have on your radar and this library can abstract away some of the challenging math behind the scenes.

Model with Aleatoric Uncertainty

- Known Unknowns
- 2 Main Model Changes
  - Add a second unit to the last dense layer before passing it to Tensorflow Probability layer to model for the predictor y and the **heteroscedasticity** or unequal scattering of data

    ```
    tf.keras.layers.Dense(1 + 1)
    ```

  - DistributionLambda is a special Keras layer that uses a Python lambda to construct a distribution based on the layer inputs and the output of the final layer of the model is passed into the loss function. This model will return a distribution for both mean and standard deviation. The 'loc' argument is the mean and is sampled across the normal distribution as well as the 'scale' argument which is the standard deviation and in this case, is a slightly increasing with a positive slope. Important to note here is that we have prior knowledge that the relationship between the label and data is linear. However, for a more dynamic, flexible approach you can use the **VariationalGaussianProcess Layer**. This is beyond the scope of this course but as mentioned can add more flexibility.

    ```
    tfp.layers.DistributionLambda(
            lambda t:tfp.distributions.Normal(
                loc=t[..., :1],
                scale=1e-3 + tf.math.softplus(0.1 * t[...,1:])

            )
    ```

- We can use different loss functions such as mean squared error(MSE) or negloglik. Note that if we decide to use the standard MSE metric that the scale or standard deviation will be fixed. Below is code from the regression tutorial for using negative

log-likelihood loss, which through minimization is a way to maximize the probability of the continuous labels.

```
  negloglik = lambda y, rv_y: -rv_y.log_prob(y)
  model.compile(optimizer='adam', loss=negloglik, metrics=[loss_metric])
```

- Extracting the mean and standard deviations for each prediction by passing the test dataset to the probability model. Then, we can call mean() or stddev() to extract these tensors.

```
yhat = prob_model(x_tst)
m = yhat.mean()
s = yhat.stddev()
```

Model with Epistemic Uncertainty

- Unknown Unknowns
- Add **Tensorflow Probability DenseVariational Layer** with prior and posterior functions. Below are examples adapted from the **Tensorflow Probability Regression tutorial notebook**.

```
def posterior_mean_field(kernel_size, bias_size=0, dtype=None):
  n = kernel_size + bias_size
  c = np.log(np.expm1(1.))
  return tf.keras.Sequential([
      tfp.layers.VariableLayer(2*n, dtype=dtype),
      tfp.layers.DistributionLambda(lambda t: tfp.distributions.Independent(
          tfp.distributions.Normal(loc=t[..., :n],
                                   scale=1e-5 + tf.nn.softplus(c + t[..., n:])),
          reinterpreted_batch_ndims=1)),
  ])
def prior_trainable(kernel_size, bias_size=0, dtype=None):
  n = kernel_size + bias_size
  return tf.keras.Sequential([
      tfp.layers.VariableLayer(n, dtype=dtype),
      tfp.layers.DistributionLambda(lambda t: tfp.distributions.Independent(
          tfp.distributions.Normal(loc=t, scale=1),
          reinterpreted_batch_ndims=1)),
  ])
```

- Here is how the 'posterior_mean_field' and 'prior_trainable' functions are added as arguments to the DenseVariational layer that precedes the DistributionLambda layer we covered earlier.

```
tf.keras.layers.Dense(75, activation='relu'),
         tfp.layers.DenseVariational(1+1, posterior_mean_field, prior_trainable),
      tfp.layers.DistributionLambda( ....
```

Additional Resources

- **Tensorflow Regression with Probabilistic Layers Tutorial**

Model Interpretability

Note: This section is included largely as a bonus and not required in the project, but it is very good to know about.

Why is Interpretability Important for EHR Datasets?

In general, as you would find for other industries you also want to be aware of biases in your model from key features and be able to identify bugs or issues that might occur.

For example, you might diagnose issues with your model where a feature like inpatient vs outpatient might have a greater weight than expected. This could be due to the nature of the type of problem you are dealing with as there are major distinctions with how inpatient and outpatient claims are handled.

Model interpretability is an issue that is often brought up as being important in fields like healthcare because of the level of scrutiny on many decisions by regulators, the public, and the healthcare community.

People want to understand how key decisions that can seriously impact lives are made and this is where practitioners often tend to use simpler models like linear models as they are easy to interpret. They want to avoid the proverbial **Black Box** model as it's important to have a clear idea of how a model is getting its predictions.

Black Box: Term often used in software when inputs go in and outputs come out of an algorithm and it is not clear how the outputs were arrived at.

However, these simpler linear models lack the complexity to handle more sophisticated cases that can have a bigger impact on society that deep learning models can potentially handle. That is why we need to use model interpretability to help remove the **Black Box** from our better, more complicated models.

Model Interpretability Methods

Model Agnostic: methods that can be used on deep learning models or traditional ml models A few methods we can use for interpreting deep learning models that are also model agnostic include:

- LIME or **Local Interpretable Model-Agnostic Explanations**
  - Can scale to large datasets
  - Can be difficult to find the right kernel
  - Can have unstable interpretations
- Shapley Values
  - A method that measures the marginal contributions of features
  - Can be computationally expensive

Shapley Values

Shapley values are based on game theory and provide the marginal contributions of features by taking the permutations of different features and then the differences between the prediction output. You can learn more about these in the links below.

Additional Resources

- **Shapley Value Explanation**
- **Integrated Gradients** - method that is specialized for deep learning models

Shapley Value Walkthrough

- Using Open Source Library Shap - **https://github.com/slundberg/shap**
- The algorithms and visualizations used in this package came primarily out of research in Su-In Lee's lab at the University of Washington, and Microsoft Research.

Steps

- Train model without Tensorflow Probability layers, for this walkthrough we will call it 'shap_model'.
- Use Kmeans clustering to summarize data. As we mentioned earlier the issue with Shapley can be the computation time and a way to reduce this is to cluster the features with Kmeans. In this example, we arbitrarily use 25 clusters but you could use a different number.

```
df_train_normed_summary = shap.kmeans(normed_train_data.values, 25)
```

- Use Shapley package's model agnostic KernelExplainer class and pass in the Shapley model and the summarized training data that has already been normalized in the previous walkthrough.

```
explainer = shap.KernelExplainer(shap_model.predict, df_train_normed_summary)
```

- Extract Shapley values from the explainer.

```
shap_values = explainer.shap_values(normed_train_data.values)
```

- Summarize the Shapley values in a plot.

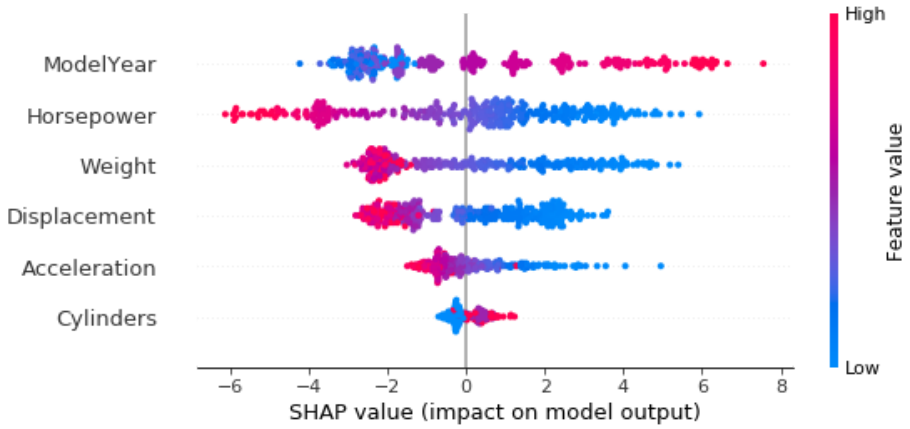
```
shap.summary_plot(shap_values[0], normed_train_data)
```

Explanation of Feature Importance Visual

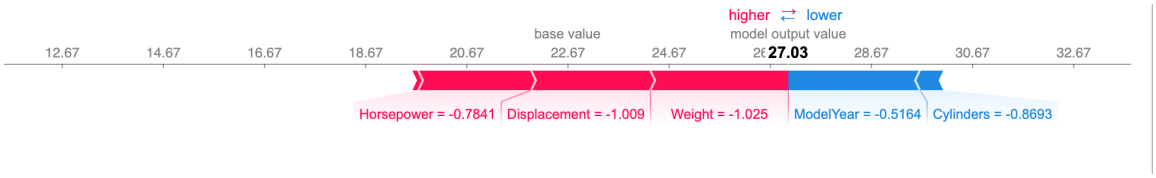
- The y-axis shows the feature importance scale from top to bottom. The sorted order of features gives the relative importance ranking of features. So in our case, model

year is the most important feature, and this shouldn't be surprising considering MPG standards increase each year.

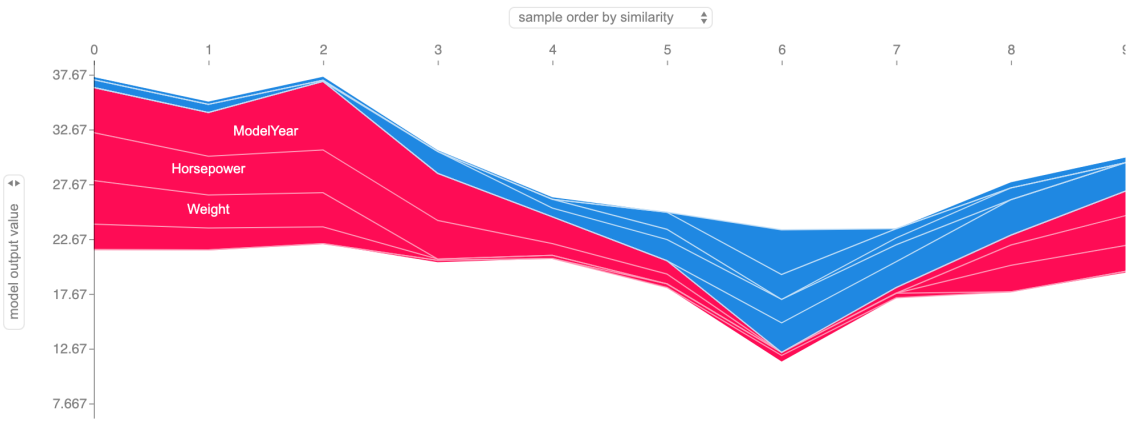
- Then, on the x-axis, you can see the Shapley values impact on model output. Feature values are either red, which stands for high values or blue which is for low values. Model year has red values that show a positive impact on model output, which in this case higher model year values yield higher MPG value output predictions.



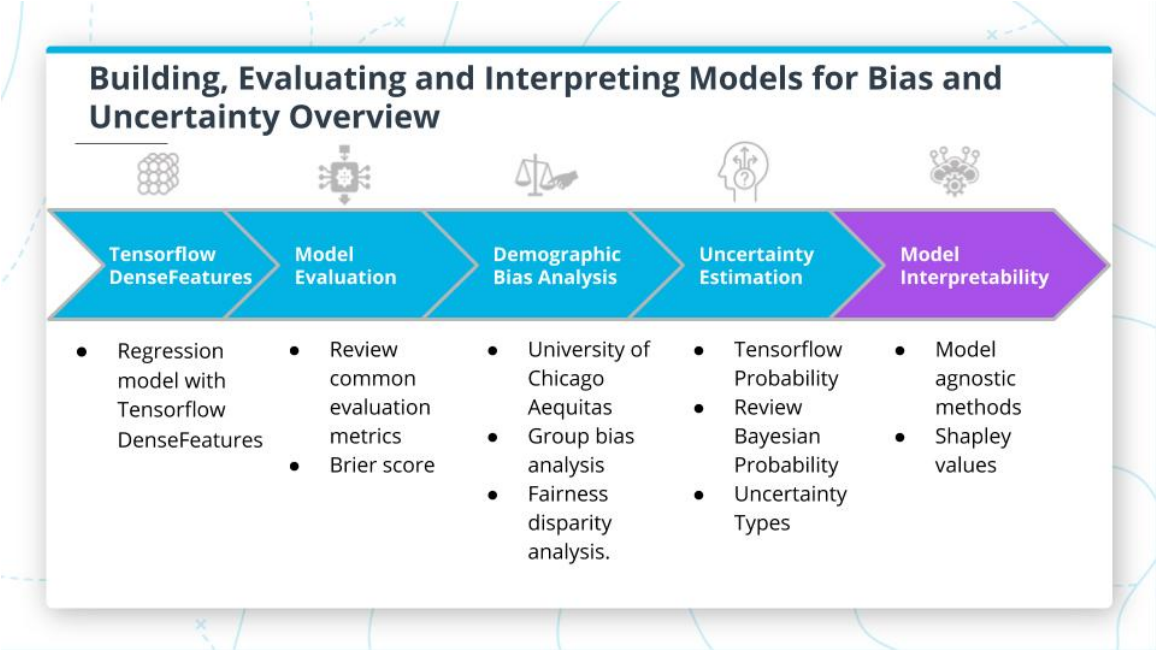
- Single point visualization - Next, we will look at a single point in this visualization. This shows 5 features change the base value towards zero. The base value is the average model output over the training dataset. The features in red displacement, horsepower and weight are pushing the label to a higher value. Whereas model year and cylinder, in this case, are pushing lower.



- Larger Sample- Lastly let's take that last visual and expand it to a sample of 10 points. Hopefully, this better illustrates how certain features are pushing the value up or down relative to a base value.



Building Models Recap



Lesson Recap

That's it! You have completed the content for this course! Well done! Let's recap what you learned in this lesson!

- You got hands-on using Tensorflow `DenseFeatures` for a model. You used `DenseFeatures` to build a simple regression model.
- You reviewed some common evaluation metrics for EHR models and then implemented Brier Scores to help evaluate your models better!
- You conducted a demographic bias analysis and used Aequitas for group bias and fairness disparity analysis to make sure that your models generalize better to real-world data.
- You used uncertainty estimation and Tensorflow Probability to build a BNN and reviewed some of the underlying concepts for Bayesian probability and types of uncertainty.
- You wrapped this lesson up by interpreting models with model agnostic methods like Shapley values.

You have learned so much in just this one lesson! Hopefully, it was both the most fun and most challenging part as well.

Lesson Key Terms

Aa Key Term	Definition

Aa Key Term	Definition
ROC	Receiver Operating Characteristic Curve or ROC curve that shows a graph of the performance of a classification model. It is the True Positive Rate Vs. False Positive Rate across different thresholds.
AUC	Area under the ROC Curve measures the entire two-dimensional area underneath the entire ROC curve.
Precision	The fraction of relevant instances among the retrieved instances
Recall	The fraction of the total amount of relevant instances that were actually retrieved.
F1	Harmonic mean between precision and recall
RMSE	Root Mean Square Error- a measure of the differences between values predicted by a model.
MAPE	Mean Absolute Percentage Error is a measure of quality for regression model loss.
MAE:	Mean Absolute Error is a measure of errors between paired observations.
Unintended Biases:	A bias that is not intentional and often is not even apparent to the creator of a model
Prior distribution	P(A)
P(A ~ Evidence)	Updating the prior with new evidence
P(B)	Posterior probability
Aleatoric Uncertainty	Otherwise known as statistical uncertainty and are known unknowns. This type of uncertainty is inherent and just a part of the stochasticity that naturally exists.
Epistemic Uncertainty	Also known as systemic uncertainty and are unknown unknowns. This type of uncertainty can be improved by adding parameters/features that might measure something in more detail or provide more knowledge.
Black Box	Term often used in software when inputs go in and outputs come out of an algorithm and it is not clear how the outputs were arrived at.
Model Agnostic	methods that can be used on deep learning models or traditional ml models