Stanford MEDICINE

Stanford | Center for Health Education

INTRO TO CLINICAL DATA STUDY GUIDE

MODULE 4 – CREATING ANALYSIS READY DATASETS FROM PATIENT TIMELINES

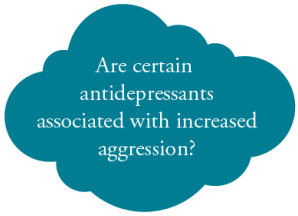
LEARNING OBJECTIVES

1. Identify the key steps in converting messy clinical data into the tabular shape used in machine learning
2. List some factors that guide the decision to include or exclude a feature
3. Describe what is meant by missing data and explain ways to address it
4. Describe the goal of feature engineering
5. Define what a knowledge graph is, and give an example of one
6. Explain how a knowledge graph can help in analyzing clinical data

CREATING FEATURES TO ANALYZE

In this conversation, we'll dive into the construction of a patient-feature matrix that is the foundation for all subsequent analyses. The **patient-feature matrix** contains data about patients in a tabular format. The data for a given patient occupies a single row. Each column is a different measurement or feature.

PATIENT	MOST RECENT BLOOD SUGAR	SMOKER?	SEX	AGE
1	101	1	F	55
2	120	0	M	12
...	...	...	...	...



PATIENT	TAKING SEROQUEL?	TAKING PROZAC?	AGGRESSIVE?
1	1	1	0
2	0	1	1
...	...	...	...

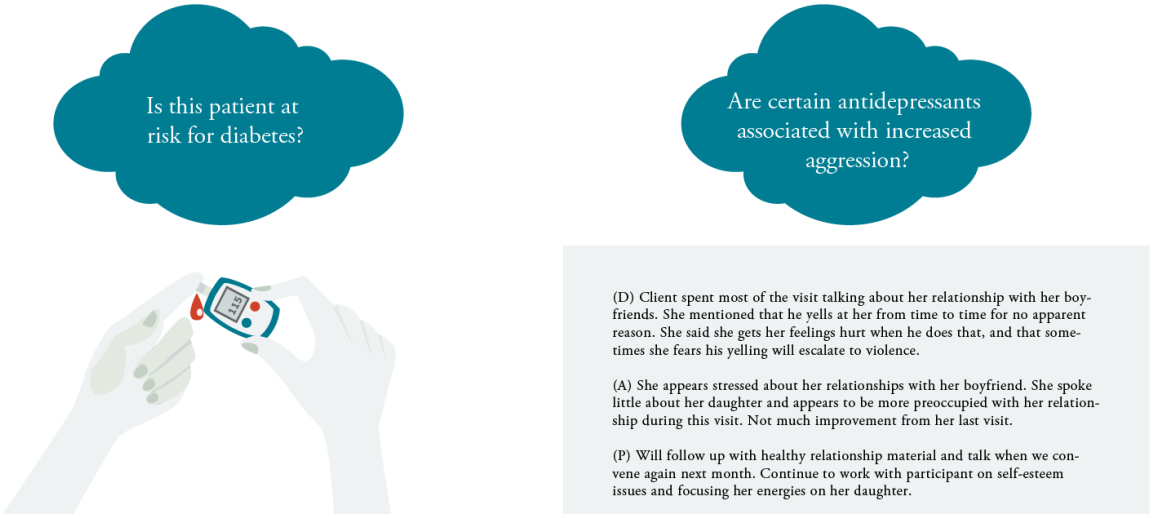
In clinical studies, the unit of observation and analysis is almost always the patient. In the data frame, each row contains all the data for a single patient. Each column contains a different feature.

Once we have decided the unit of observation, and analysis, we turn to determining which features to extract for the data.

It is best to start with all the features. If there are constraints on computing resources, then you may need to remove features to reduce the size of the dataset. Some modern machine learning methods can automatically remove those features that contribute the least to the accuracy of the model.

Important: You may need to remove some sensitive features for patient privacy.

We can use subtle information implicit in the data to help us craft features. Metadata, data that refers to other data, can be quite informative.



Example: We need to determine who is a diabetic. Ideally we would look at the *results* of a test such as the HbA1C. However, we can also use the counts of orders of laboratory tests that have something to do with measuring glucose instead of the actual results of those tests. So, we used the metadata -- the number of times a test related to measuring glucose is ordered -- and some prior knowledge that diabetes mellitus is a disease where glucose levels get messed up to craft a **feature (percentage of tests that are about glucose)** which informs us whether someone is a diabetic or not.

Usually such features are **created**, or **engineered**, by using some prior knowledge. It is also possible to learn such features via computation.

Healthcare data can be structured or can be unstructured.

Making datasets from structured sources:

1. Accessing structured data
2. Standardizing features
3. Dealing with too many features
4. Dealing with missing data
5. Constructing new features

Structured data, generally reside in database tables. Databases are queried using SQL (Structured Query Language) and the results can be loaded into systems for analysis. Data in different tables may be linked using a database operation called a "join". The data may need to be reshaped into a useable format.

It is common to standardize features, which transforms all features into a common numerical range. The process of standardizing is sometimes called **normalizing**. Standardization facilitates later analysis by reducing the effect of values that are extremely large or extremely small relative to other values in the dataset.

PATIENT	DX 993.4	BLOOD SUGAR	AGE IN DAYS	...
1	0	120	11315	
2	2	120	32000	
30	0	110	6003	
46	0	120	13500	
54	0	130	522	

Different scales will throw off many analyses.

PATIENT	DX 993.4	BLOOD SUGAR	AGE	...
1	0	0.92	0.34	
2	1	0.92	1	
30	0	0.84	0.18	
46	0	0.92	0.36	
54	0	1	0.01	

SCALE

$$X'_j = \frac{X_j - \min(X_j)}{\max(X_j) - \min(X_j)}$$

A commonly used transformation rescales each column so it spans the same range, often 0 to 1. Another transformation is such that the column has an arithmetic mean of 0 with a standard deviation of 1.

Reasons why you might not want to use all of the features:

- Some features might be useless
- Missingness: A feature might be missing for most patients
- Sparsity: A large number of features are missing for a given patient
- Redundancy: Feature1 might be highly correlated with Feature2

- Speed: Large number of features can slow the analysis
- Privacy: Saving more features increases the chance of violating patient privacy

Low-prevalence, low-variance features are good candidates for removal.

Another way to reduce features is to combine features using domain knowledge. A benefit of such aggregation is that it may remove some idiosyncrasies of how individual clinical sites code features, making cross-site comparison easier. The aggregation step requires accurate representations of domain knowledge in the form knowledge graphs.

It is also possible to use mathematical operations to detect and use patterns in the data. Many of these methods, such as principal components analysis (PCA), use linear algebra. The benefits of using such techniques is that they are domain-independent and do not hinge on specific medical knowledge.

The main drawback of mathematically combining existing features, is that it makes the derived feature difficult to interpret. The new derived feature set may enable accurate predictions, but the features that contribute to the prediction may not have understandable clinical interpretation.

When you are considering reducing the number of features:

- Think about whether the distinctions reflected in a feature are relevant to your question
- The more flexible the model you are using in the later analysis stage, the less benefit aggregation will provide. Regressions will benefit more than gradient boosted trees
- Reducing the required computational resources provides benefits independent of the choice of model

MISSING VALUES

When we convert a patient timeline view of the data into a patient-feature matrix, naturally some entries in this matrix will be missing.

Missing Data: In prospective studies, when a value for a data element is missing, it is reasonable to assume that it should have been recorded but was not. However, in data that are a byproduct of routine care, just because there is a place in a user constructed patient-feature matrix, does not mean that the value **should** have been recorded

The absence of a specific value in a column of patient feature matrix could mean three things:

1. The value should have existed, but does not (the usual meaning of missing data)

2. The value not being present is an artifact of adopting a tabular view of the data
3. The value could have existed, but was deemed unnecessary to collect

Absent values create problems for analysis in two ways:

1) From how they are reported
2) If they were truly 'missing data', then from how they are imputed

Some systems allow for a special data element, often written as "NA" or "null", to represent a missing value. Other systems might use numerical values that are outside the range of possible values for that feature, such as 0 or -999, to denote a missing value.

Removing patient records with missing values a tempting and simple solution, but it often creates problems. Removing patients with a missing value would tend to remove under represented patients from dataset. This would bias our analysis.

Dealing with missing values

One widely used method is to impute the missing values. **Imputation** is a kind of prediction that fills in the missing values based on other information in the dataset.

A simple imputation method, called **column mean imputation**, replaces the missing value with the mean of the known values in the same column. This assumes that the variable's values in the other rows of that column have information about the missing value, which is often not true in medicine.

We can use **values in other columns of the same patient** to improve the imputation procedure. This is usually better than column imputation which considers only values in one column at a time. We are using expected correlations among different features of the same patient to infer the value of the missing feature.

A procedure called **k-nearest neighbors imputation** fills in a missing cell by looking for patients who are similar to the current patient on the basis of other features, and then uses those known values to impute the missing one.

A modern method called **multiple imputation**, repeatedly invokes imputation to create multiple versions of the data, which can be analyzed to provide an estimate of the variance in the imputed values.

Making a decision on whether **to remove missing values or to impute:**

- If a variable is mostly measured with only a few missing values, then you should consider imputation
- If a particular variable has mostly missing values, then you should consider dropping the variable. That variable does not contribute useful information for most patients, and we would have to impute the values for most patients.

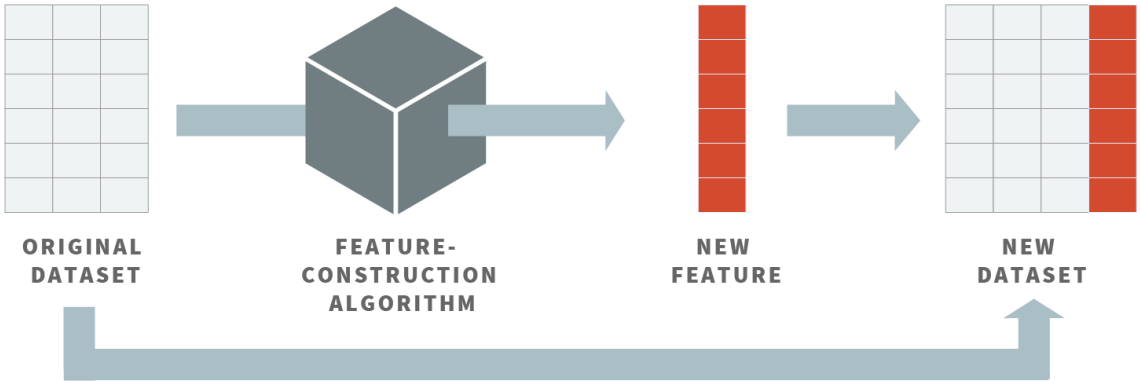
If the distribution of missing values is in middle-zone between these alternatives, some experts have advised adding indicator variables to mark which values have been imputed, but this recommendation has been disputed by others.

From a practical perspective, it may be better to use an analysis method that naturally handles missingness rather than imputing the missing values.

Finally, think about how important the feature with missing data is to your question. Could you avoid imputation by answering the question without considering that variable?

CREATING NEW FEATURES

As we convert a patient timeline view of the data into a patient-feature matrix, we can also perform simple operations on the source data **to create new features**. Such construction of new features is called "**feature engineering**".



Constructed features are transformations of the original features or their combinations. Simple models with well-engineered features can perform better than fancy models with original features.

Examples of engineered features

Clinical scoring systems, simple formulas that combine values found in the EMR, are great examples of engineered features. The body mass index is a relatively simple example of a scoring system that allows us to estimate the severity of how over or underweight someone is.

Other scoring systems quantify the overall burden of multiple diseases; often called the **comorbidity burden**. They are typically used to account for overall patient illness in analyses and avoid comparing sick people to healthy people.

Among other examples: Create proxy features for a patient's socioeconomic status from their zip code, and the number of EMR records they have, scaled by a measure of their overall health discussed above. Infer unrecorded conditions, such as smoking status, by looking for the presence of keywords in text, such as “cigarette”. In other cases, could look for specific combinations of drugs and procedures. Can also use clinical knowledge to guide feature engineering.

General Advice for Feature Engineering:

- Think about what features might be important but are not directly measured
- Take advantage of pre-validated clinical scoring systems
- When creating a new feature, consider including counts, differences, change over time, and ratios of existing measurements
- Lean towards creating new features using some clinical knowledge and creativity
- Balance the benefit from building new features against the effort used to create them

Deep learning: A new method in machine learning that uses more than conventional amounts of raw data and builds the needed features without domain knowledge

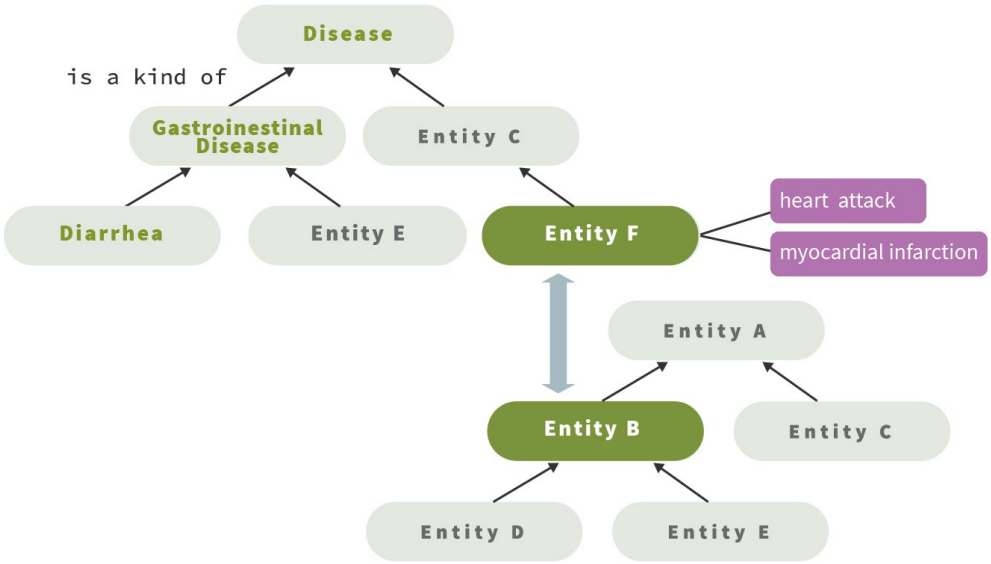
Creating analysis ready datasets from patient timelines

- Structured data in database tables can be transformed into analysis ready datasets called patient feature matrix
- The number of features in the patient feature matrix can be reduced by aggregating features using domain knowledge, or by using mathematical techniques such as principal component analysis
- Missing data can be removed, or imputed with different levels of methodological sophistication
- Consider creating additional features from the original data

KNOWLEDGE GRAPHS

A **knowledge graph** is a declaration of what entities exist in a domain and the relationships among them. It is also referred to as an ontology. Ideally represented in digital form.

A common problem when working with clinical data is that there are multiple ways to express the same concept. Knowledge graphs can help because they **explicitly represent synonyms of entities** and the **relations among different entities**.



So what exactly is in a knowledge graph?

1. They contain **entities**.
2. There are sets of **equivalent names** for those entities, or synonyms.
3. There are **relations** between the entities. A very common relation represents "is a kind of"
    - This 'kind of' relationship in Knowledge graphs is the most important because it codifies what is a kind of what in the medical domain. Entities will inherit properties from the entities they are a kind of.
4. Contain **links to other knowledge graphs**. This can provide a consistent way to refer to entities across different data sources

Knowledge graphs are extremely useful when querying clinical data. They help identify different terms with the same meaning.

There are hundreds of knowledge graphs available in medicine and in biological research. If you want to explore the many knowledge graphs available, check out the BioPortal from the National Center for Biomedical Ontology at Stanford.

Important Knowledge Graphs:

- International Classification of Diseases (ICD-10. ICD-9)
- The Current Procedural Terminology (CPT)
- RxNorm and RxNav provided by the US National Library of Medicine (NLM)
- Anatomic Therapeutic Chemical (ATC) Classification System
- The Logical Observation Identifiers Names and Codes (LOINC)

The Unified Medical Language System's metathesaurus, or the UMLS metathesaurus, is a union of over 140 knowledge graphs. It contains all of the **knowledge graphs** we have just mentioned, along with declarations of **relationships *between* these knowledge graphs**.

Questions to evaluate a knowledge graph:

1. What are the **entities** the knowledge graph has, and what is the basis of classification?
2. What **words** are used to name the entities in the graph? Are there synonyms and alternative names?
3. Is it **mapped** to other knowledge graphs? How 'connected' is a knowledge graph with other knowledge graphs?

Aside from these three principled questions, there are some practical approaches to assessing a knowledge graph. For example, given data from an EMR, count the number of terms from each knowledge graph that are mentioned in EMR text documents. In addition, if the knowledge graph is too big, with millions of terms, you can use the counts of term occurrences to help decide which terms to keep.

In summary, knowledge graphs are large, highly curated collections of medical **entities**, alternative **names** of those entities, and **relationships** between them. They are an extremely important source of medical knowledge for creating features and processing clinical text.