GENERAL ASSEMBLY

Intro to Data Cleaning

Patrick D. Smith

Lead Instructor, DSI DC

LEARNING OBJECTIVES

- Inspect data and data types
- Cleaning up data

Introduction to Data Cleaning

Intro to Data Cleaning

Since we're starting to get pretty comfortable with using pandas to do EDA, let's add a couple more tools to our toolbox.

Conceptually: what do you look for, and how do you stay organized?

Intro to Data Cleaning

Since we're starting to get pretty comfortable with using pandas to do EDA, let's add a couple more tools to our toolbox.

Conceptually: what do you look for, and how do you stay organized?

There's no magic formula, but we'll go over some common cleaning operations.

Intro to Data Cleaning

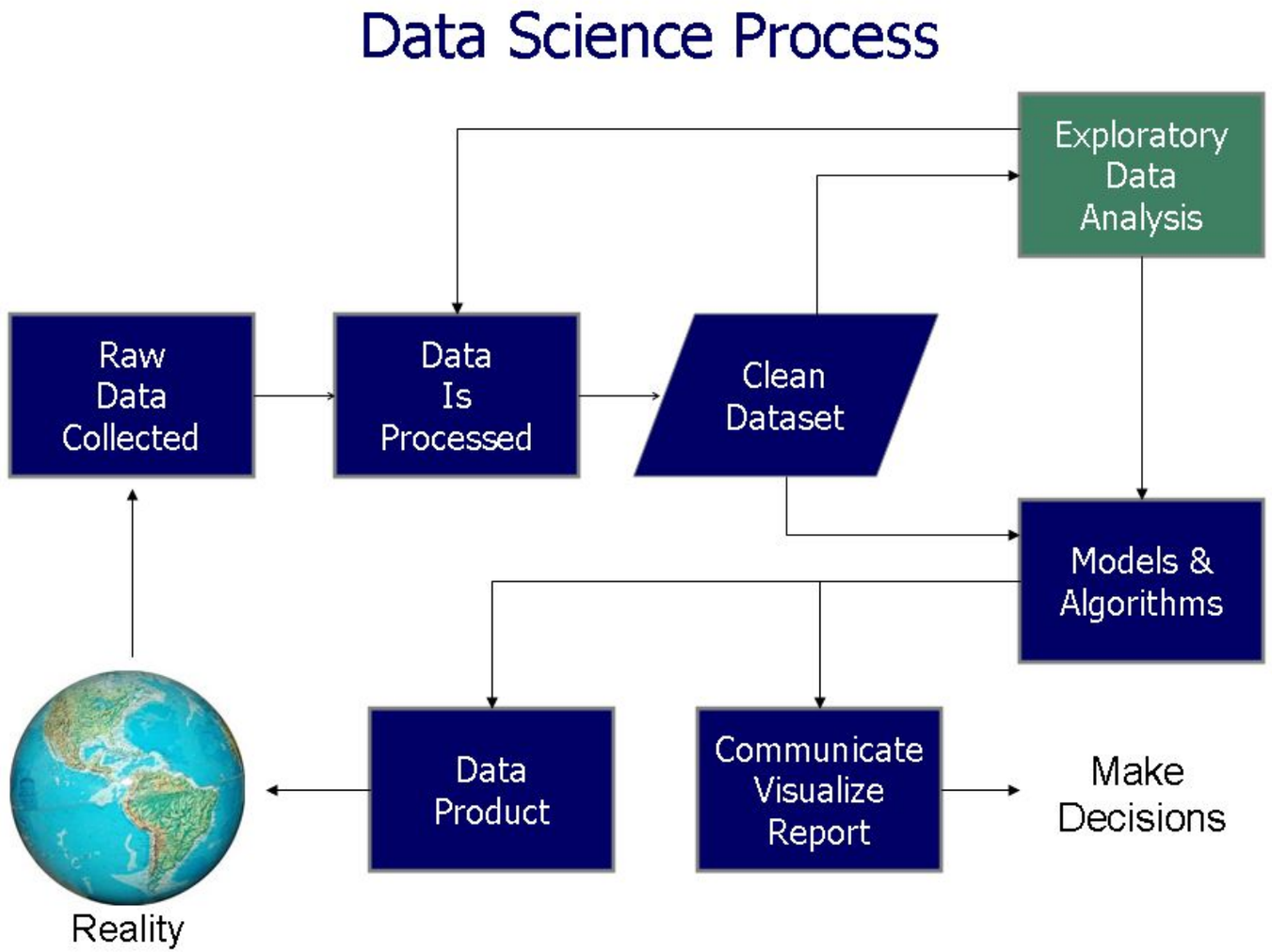
Since we're starting to get pretty comfortable with using pandas to do EDA, let's add a couple more tools to our toolbox.

Conceptually: what do you look for, and how do you stay organized?

There's no magic formula, but we'll go over some common cleaning operations.

Reproducibility matters, so document! For high-level planning and documentation, ***data flow diagrams*** are helpful.

Intro to Data Cleaning



Intro to Data Cleaning

Technically: once you know what you want to do, how do you do it in pandas?

Pandas has many functions to help process and manipulate your data - we'll take a look at some:

- .dtypes is the data type attribute of numpy/pandas objects.
- df.apply() applies a function along any axis of a DataFrame.
- pandas.Series.value_counts returns a Series containing counts of unique values. Excludes NaN values.

The main data types stored in pandas objects are:
float, int, bool, datetime64, datetime64, timedelta, category, and object

Intro to Data Cleaning

Common Steps in Cleaning Data:

- Drop Outliers
- Normalization
- Relabeling
- Decoding
- Handling Null Values
- Binarization

.

Let's take a look at an example of how to inspect data types

Intro - Applying functions functions to dataframes

Intro to Data Cleaning

Generally, df.apply(), will apply a singlular function to every cell of the dataframe you use it with.

Conversely: df.map(), is available when you only want to work with a single dimension of your dataset, ie: df['a'].map(my_func)

Intro - .value_counts()

Intro to Data Cleaning

Why is this important? Basically, this tells us the count of unique values that exist. It's helpful to identify anything unexpected. Looking at value_counts(), per series, can give us a quick overview of values expressed in our data.

- Strings inside of mostly numeric / continious data
- Non-numeric values
- General counts of values that we might expect to see
- Most common / least common values

Independent Practice

Intro to Data Cleaning: Independent Practice

- Use the sales.csv data set, we've seen this a few times in previous lessons
- Inspect the data types
- You've found out that all your values in column 1 are off by 1. Use df.apply to add 1 to column 1 of the dataset
- Use .value_counts to count the values of 1 column of the dataset

Bonus

- Add 3 to column 2
- Use .value_counts for each column of the dataset

Intro to Data Cleaning: Independent Practice

- Add an extra column to your dataframe that is a copy of an existing column with continious data
  - Randomly change the value of continious data cells within it to the following:
    - NaN
    - A blank string
    - A numeric string
    - The same value

Report value_counts post-"random data troll" processing. Does it seem random?

- Convert blank strings and NaN values to float(0)
- Convert numeric strings to floats with 2f precision
- Divide by 2 if cell value is prime, use remainder as value

Conclusion