

[Source](#)

# Welcome to Week 2 Lecture 2!

Data Science in Python &  
Machine Learning



# Week 2 CORE Assignments

These **MUST** be submitted by Sunday February 6th:

- 1) Loading & Filtering Data
- 2) Titanic Data
- 3) Project 1 - Part 2

# Resources of Note on the LP

- [Making Data Tidy \(optional assignment\)](#):
  - The optional assignment is hard and I recommended skipping it.
  - BUT there is an [awesome Pydata Pandas cheat sheet](#) linked at the bottom!
- [Prep For Belt Exam](#) Section
  - Between Week 2 and Week 3 is the [mock belt exam](#).
  - After this week's assignments, you should know how to tackle part 1 of 2!
  - **We will go over the mock belt exam in code reviews next week.**

# Resources of Note: Cohort Notes Repo

**New Cohort Notes Repository** [\[Link\]](#) (Link is also on Helpful Links sheet)

- Same content as class, just organized into week/lecture folders.

## Helpful Links

### BOOTCAMP LINKS TO BOOKMARK

[Learn Platform](#) - Dashboard

[Class Google Cal](#)

[Stack 1: DS Fundamentals](#)

[TA Support Good](#)

Discord Channel: [01](#)

[Cohort Notes Repository](#)

The screenshot shows the GitHub interface for the repository 'sensei-jirving / Online-DS-PT-01.24.22-cohort-notes'. The repository is public and has 12 commits. The commit history is as follows:

Commit Message	Time Ago
Added folder infrastructure	43 minutes ago
Created using Colaboratory	4 minutes ago
Created using Colaboratory	42 minutes ago
Added folder infrastructure	43 minutes ago
Added folder infrastructure	43 minutes ago
Added folder infrastructure	43 minutes ago
Added folder infrastructure	43 minutes ago

# Code Reviews Next Week - Mock Belt Exam

[illegible]

# Learning Goals:

By the end of this class you will be able to:

1. Identify the types of data cleanliness issues to fix.
2. Find and remove irrelevant columns
3. Find and drop duplicated rows
4. Find and correct badly-formatted/obviously-incorrect data.
5. Quantify missing data and select from a variety of strategies to fill or remove it.

# Why do we need to clean our data?



Garbage in, garbage out.



# When do we need to clean our data?

Always!

~ 80% of any data science project is data cleaning!

# Data Issues to Address/Clean

- Duplicate data
- Irrelevant data
- Incorrect data types
- Inconsistent Values
- Syntax/Spelling Errors

# Duplicate data

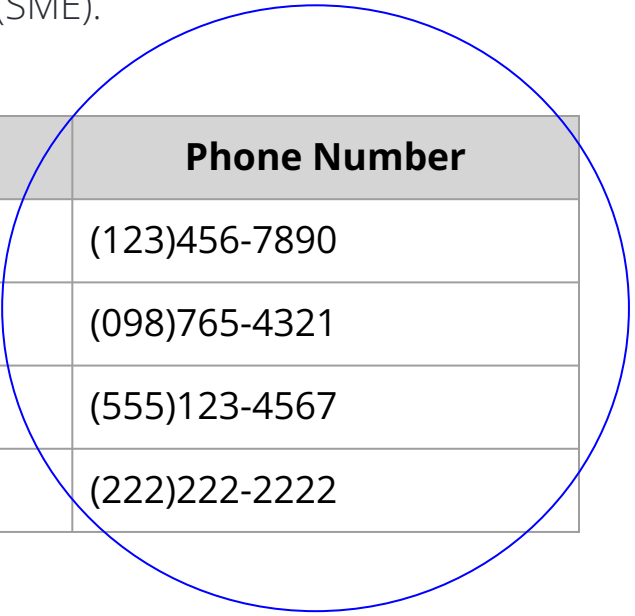
- Duplicate **rows** should be removed.

Height	Age	Phone Number
70	31	(123)456-7890
72	45	(098)765-4321
73	40	(555)123-4567
72	45	(098)765-4321

# Irrelevant data

- We want to remove data if it is unnecessary or unrelated to our task.
  - When in doubt, consult a subject matter expert (SME).

Height	Age	Phone Number
70	31	(123)456-7890
72	45	(098)765-4321
73	40	(555)123-4567
71	42	(222)222-2222



# Examples of Irrelevant data

- Columns with values that are **unique in every row** (you could consider using this as an index)
- Columns with **values that are all identical**
- An **Unnamed: 0** column
- Other columns that are just unnecessary for your task

# Data Types

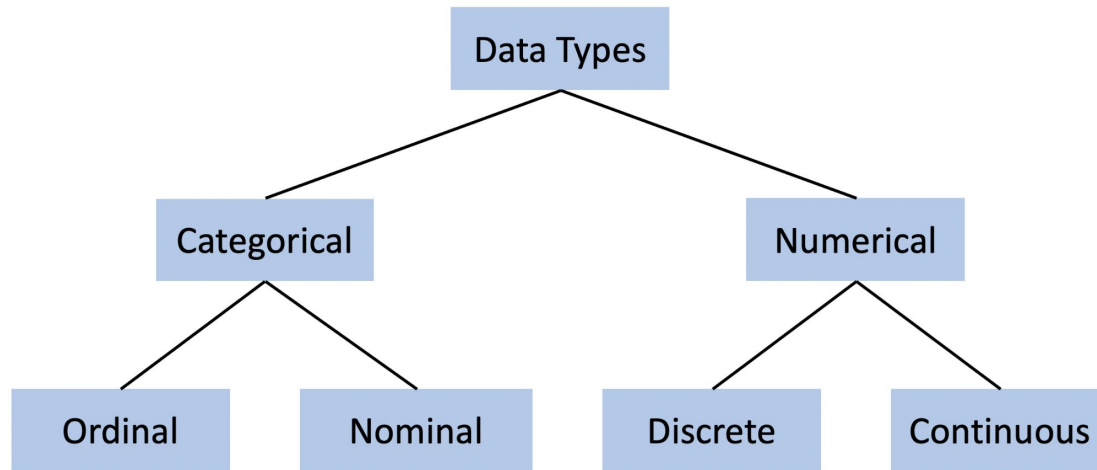


Image Source: <https://bookdown.org/ejvanholm/Textbook/displaying-data.html>

Example of a numeric and/or a categorical variable in data science?

<u>Numerical</u>	Type in chat or annotate here:	<u>Categorical</u>

# Incorrect Data types

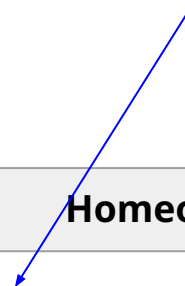
- An unexpected data type could be a clue to bad data!
  - Numerical data should be an **int** or **float**
  - Be careful, numbers can represent ***categories***



# Data types

These are just  
categories, NOT  
numeric values

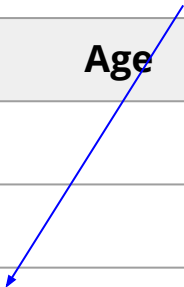
Height	Age	Homeowner
70	31	0
72	45	1
73	40	1
72	45	0



# Data types

If even one value is typed in as a string, the column will NOT be recognized as numbers

Height	Age	Homeowner
70	31	0
72	45	1
73	forty	1
72	45	0



You can correct this manually by using:

```
df.loc[2, 'Age'] = 40
```

# Unit inconsistencies

- If measurements are in different units (for example, km vs. miles) we should ensure consistency in the data.

# Unit inconsistencies

Height	Age	Pet
70	31	cat
73	40	dog
6	43	dog
68	32	dog

- Is the person really only 6 inches tall?
- Is it completely invalid data?
- Is it possible the data was entered in the wrong unit?

Data cleaning requires **judgement calls** which can be very frustrating!!

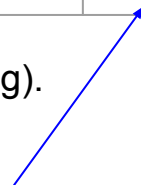
# Formatting and Syntax

# Categorical Data

Height	Age	Pet
70	31	Cat
72	45	cat
73	40	dog
72	45	Dogs
68	32	dogg
71	35	cat

We want TWO category options for pet (either cat or dog).

How many categories does this data table show?



# Common Syntax or spelling errors

- Upper and lowercase letters
- Plural forms
- Spelling errors or typos
- Abbreviations
- Extra white spaces at the beginning or end of strings or column names.

# Outliers

Too big? Too Small? Seems unlikely?



# Outliers

- Outliers are values that are extremely higher or lower than all of the other values in that feature.
- If you see outliers in your data, it might be due to unit inconsistencies (check first)
- In general, outliers should not be removed unless you have reason to believe it is an error in the dataset.

# Outliers: Unusual values

- If your dataset is small enough, outliers can be identified by just eyeballing the value.  
2.1, 3.7, 1.5, 3.0, 156.1, 2.6, 1.9
- For larger datasets, **visualizations** are a quick way to spot outliers.



We will learn data visualization next week!



Always Fix Missing Data!!

# Missing Data

Height	Age	Pet
70	31	cat
73	40	dog
72	NaN	dog
68	32	NaN

- Why is the value missing?
- Are there any patterns in missing data?
- Does a missing value under Pet simply mean “no pet”?
- Was pet left blank because the person has a turtle which wasn't a choice?

Data cleaning requires **judgement calls** which can be very frustrating!!

# Missing Data

- Missing values are very common and problematic in our data. There is no one right way to handle missing data. A few strategies include:
  - Drop rows
  - Drop columns
  - Impute values

# Dropping Missing Data

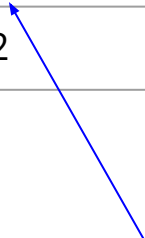
- If there are not very many missing values and they are missing completely at random, the easiest solution is to drop the rows or columns with missing values.
- A good rule of thumb is to limit dropped data to 5%

# Imputing Missing Data

- Imputing means filling in the missing values based on some criterion. There are many methods to do this:
  - Simple imputation including mean, median, or mode imputation
  - More Advanced Imputation: Using a predictive model to impute the values

# Imputing Missing Data

Height	Age	Pet
70	31	cat
73	40	dog
72	NaN	dog
68	32	cat



If using mean, the missing value would be replaced with 34 (or 34.3)



# Imputing Missing Data

- NOTE about imputation:
  - The Learn Platform curriculum introduces 'ffill', 'bfill', and 'linear' methods of imputing using `df.fillna()`.
  - Only consider this if the data is ordered in some way and those make sense. One example might be if each row is a date or time and they are organized chronologically.
  - Generally use 'median', 'mean', 'mode', or a constant value depending on the circumstances.

# ¥Our turn!

## Data:

Here is the dataset you will be working with:

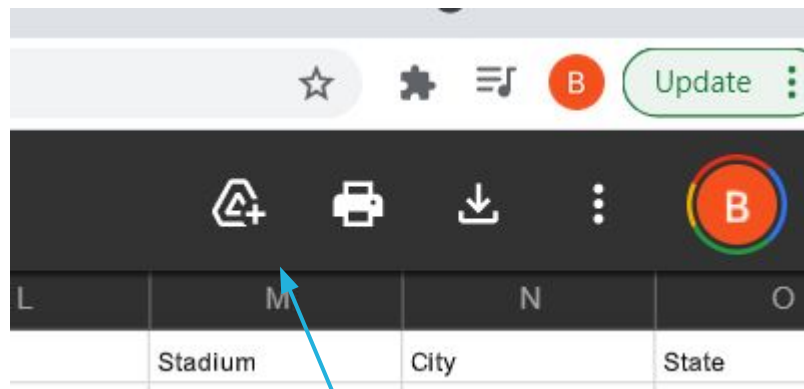
[Superbowl data set](#)

## Notebook:

Go to this link and **make a copy!**

[Colab Notebook](#)

# Happy Coding!



Open the link and then add the shortcut to your drive (or download/upload)

# How-To Slides

## Python Reference

# Dropping a Column

An entire column can be dropped using:

```
df.drop(columns = 'Phone Number', inplace = True)
```

**OR**

```
df = df.drop(columns = 'Phone Number')
```

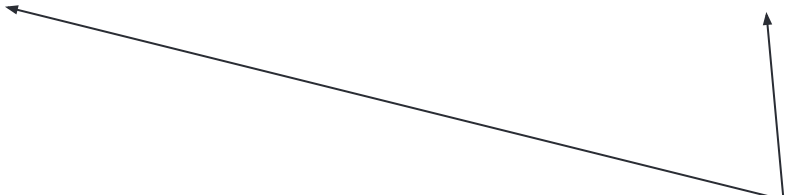
# Dropping Multiple Columns

```
df.drop(columns = ['Phone Number', 'Unnamed: 0'], inplace = True)
```


OR

```
df = df.drop(columns = ['Phone Number', 'Unnamed:0'])
```

List the  
names of  
columns  
to drop in  
square  
brackets



# Duplicate data: in Python

- Check for any duplicated data in Python using: `df.duplicated().any()`
- Check to see which data is duplicated: `df[df.duplicated(keep = False)]`
  - keep : {'first', 'last', False}, default 'first'**  
Determines which duplicates (if any) to mark.
    - **first** : Mark duplicates as **True** except for the first occurrence.
    - **last** : Mark duplicates as **True** except for the last occurrence.
    - **False** : Mark all duplicates as **True**.

A Filter!
- Drop duplicates in Python using: `df.drop_duplicates(inplace = True)`

# Data types: in Python

Check data types in Python using:  
`df.dtypes`

Or:  
`df.info()`

If there is a data dictionary, you can validate your data types with that! Otherwise, use your best judgement.

# Syntax or spelling errors: in Python

Check for counts of each categories using:  
`df[ 'column' ].value_counts()`

And look for any values that look weird/should be interpreted as the same things!



# Replace Incorrect Data

## As a list

```
df.replace(['dogg', 'Dogs'], ['dog', 'dog'], inplace = True)
```

## As a dictionary

```
df.replace({'dogg': 'dog', 'Dogs', 'dog'}, inplace = True)
```

# Identifying Missing Data

`df.info()` → Tells the number of values in each column

`df.isna().sum()` → Sums the number of nas in each column

`df[df.isna().any(axis=1)]` → **Filter!** To display only rows with na values

# Dropping Missing Data: in Python

Dropping rows can be done in Python using:

```
df.dropna(inplace = True)
```

Or to drop columns:


```
df.drop(columns = ['col1', 'col2'],  
        inplace = True)
```

# Imputing Missing Data: in Pandas

This can be done in Pandas using:

```
df[col].fillna(value = 6, inplace = True)
```

You provide a specific  
value such as 6



Or you can fill with an aggregate value:

```
df[col].fillna(value = df[col].mean(), inplace = True)
```