INT 303 BIG DATA ANALYTICS

Lecture: Data Grammar

Jia WANG

Jia.wang02@xjtlu.edu.cn

Xi'an Jiaotong-Liverpool University

西交利物浦大學

BACKGROUND

So far, we've learned:

Lecture1	• What is Data Science?
Lecture1&2	• The Data Science Process
Lecture2	• Data: types, formats, issues, etc.
Lecture2	• Visualization (briefly)
This lecture	• How to quickly prepare data
Future lecture	• How to model data

BACKGROUND

- The Data Science Process:

LECTURE OUTLINE

- Exploratory Data Analysis (EDA):
  - Without Pandas (part 1) – These slides
  - With Pandas (part 2) – Mostly Jupyter Notebook
- Data concerns (part 3) – These slides

EXPLORATORY DATA ANALYSIS (EDA)

Why?

- EDA encompasses the "*explore* data" part of the data science process
- EDA is crucial but often overlooked:
  - If your data is bad, your results will be bad
  - Conversely, understanding your data well can help you create smart, appropriate models

EXPLORATORY DATA ANALYSIS (EDA)

What?

1. Store data in data structure(s) that will be convenient for exploring/processing (Memory is fast. Storage is slow)
2. Clean/format the data so that:
   - Each row represents a single object/observation/entry
   - Each column represents an attribute/property/feature of that entry
   - Values are numeric whenever possible
   - Columns contain atomic properties that cannot be further decomposed*

* Unlike food waste, which can be composted. Please consider composting food scraps.

EXPLORATORY DATA ANALYSIS (EDA)

What? (continued)

3. Explore **global** properties: use histograms, scatter plots, and aggregation functions to summarize the data
4. Explore **group** properties: group like-items together to compare subsets of the data (are the comparison results reasonable/expected?)

This process transforms your data into a format which is easier to work with, gives you a basic overview of the data's properties, and likely generates several questions for you to follow-up in subsequent analysis.

EDA: WITHOUT PANDAS

- Say we have a small dataset of the top 50 most-streamed Spotify songs, globally, for 2019.

EDA: WITHOUT PANDAS

top50.cs

- Each row represents a distinct song. The columns are:
- **ID:** a unique ID (i.e., 1-50)
- **TrackName:** Name of the Track
- **ArtistName:** Name of the Artist
- **Genre:** the genre of the track
- **BeatsPerMinute:** The tempo of the song.
- **Energy:** The energy of a song - the higher the value, the more energetic.
- **Danceability**: The higher the value, the easier it is to dance to this song.
- **Loudness**: The higher the value, the louder the song.
- **Liveness**: The higher the value, the more likely the song is a live recording.
- **Valence**: The higher the value, the more positive mood for the song.
- **Length**: The duration of the song (in seconds).
- **Acousticness**: The higher the value, the more acoustic the song is.
- **Speechiness**: The higher the value, the more spoken words the song contains.
- **Popularity**: The higher the value, the more popular the song is.

EDA: WITHOUT PANDAS

top50.csv

	TrackName	ArtistName	Genre	BeatsPer Minute	Energy	Danceability	Loudness	Live ness	Valence	Length	Acousticness	Speechi ness	Popularity
1	Senorita	Shawn Menc	canadian pop	117	55	76	-6	8	75	191	4	3	79
2	China	Anuel AA	reggaeton flow	105	81	79	-4	8	61	302	8	9	92
3	boyfriend (w	Ariana Granc	dance pop	190	80	40	-4	16	70	186	12	46	85
4	Beautiful Pec	Ed Sheeran	pop	93	65	64	-8	8	55	198	12	19	86

- **Q1:** What are some ways we can store this file into data structure(s) using regular Python (not the Pandas library).

EDA: WITHOUT PANDAS

top50.csv

	TrackName	ArtistName	Genre	BeatsPer Minute	Energy	Danceability	Loudness	Live ness	Valence	Length	Acousticness	Speechi ness	Popularity
1	Senorita	Shawn Menc	canadian pop	117	55	76	-6	8	75	191	4	3	79
2	China	Anuel AA	reggaeton flow	105	81	79	-4	8	61	302	8	9	92
3	boyfriend (w	Ariana Granc	dance pop	190	80	40	-4	16	70	186	12	46	85
4	Beautiful Pec	Ed Sheeran	pop	93	65	64	-8	8	55	198	12	19	86

- **Possible Solution #1:** A 2D array (i.e., matrix)

Weaknesses:

- What are the row and column names? Need separate lists for them – clumsy.
- Lists are O(N). We'd need 2 dictionaries just for column names

data = [][]
col_name -> index
index -> col_name

EDA: WITHOUT PANDAS

top50.csv

	TrackName	ArtistName	Genre	BeatsPer Minute	Energy	Danceability	Loudness	Live ness	Valence	Length	Acousticness	Speechi ness	Popularity
1	Senorita	Shawn Menc	canadian pop	117	55	76	-6	8	75	191	4	3	79
2	China	Anuel AA	reggaeton flow	105	81	79	-4	8	61	302	8	9	92
3	boyfriend (w	Ariana Granc	dance pop	190	80	40	-4	16	70	186	12	46	85
4	Beautiful Pec	Ed Sheeran	pop	93	65	64	-8	8	55	198	12	19	86

-
-
-

Possible Solution #2: A list of dictionaries

list

Item 1	=	{“Track.Name”: “Senorita”, “Artist.Name”: “Shawn Mendes”, “Genre”: “Canadian pop”, …}
Item 2	=	{“Track.Name”: “China”, “Artist.Name”: “Anuel AA”, “Genre”: “reggaetón flow”, … }
Item 3	=	{“Track.Name”: “Ariana Grande”, “Artist.Name”: “boyfriend”, “Genre”: “dance pop”, … }

EDA: LIST OF DICTIONARIES

Possible Solution #2: A list of dictionaries

```
f = open("../data/top50.csv", encoding = "ISO-8859-1")
column_names = f.readline().strip().split(",")[1:] # puts names in a list
cleaned_column_names = [name[1:-1] for name in column_names]
cleaned_column_names.insert(0, "ID")

dataset = []

# iterates through each line of the .csv file
for line in f:
    attributes = line.strip().split(",")

    # constructs a new dictionary for each line
    dataset.append(dict(zip(cleaned_column_names, attributes)))
```

From lecture3.ipynb

EDA: LIST OF DICTIONARIES

Possible Solution #2: A list of dictionaries

- **Q2:** Write code to print all songs (Artist and Track name) that are longer than 4 minutes (240 seconds):

```
for song in dataset:
    if int(song["Length."]) > 240:
        print(song["Artist.Name"], "-", song["Track.Name"], "is", song["Length."],"seconds long")
```

From lecture3.ipynb

EDA: LIST OF DICTIONARIES

Possible Solution #2: A list of dictionaries

- Q3: Write code to print the most popular song (artist and track) – if ties, show all ties.

```
max_score = -1
most_populars = set()
for song in dataset:
    if int(song["Popularity"]) > max_score:
        most_populars = set([str(song["Artist.Name"] + "-" + song["Track.Name"])])
        max_score = int(song["Popularity"])
    elif int(song["Popularity"]) == max_score:
        most_populars.add(str(song["Artist.Name"] + "-" + song["Track.Name"]))
print(most_populars)
```

From lecture3.ipynb

EDA: LIST OF DICTIONARIES

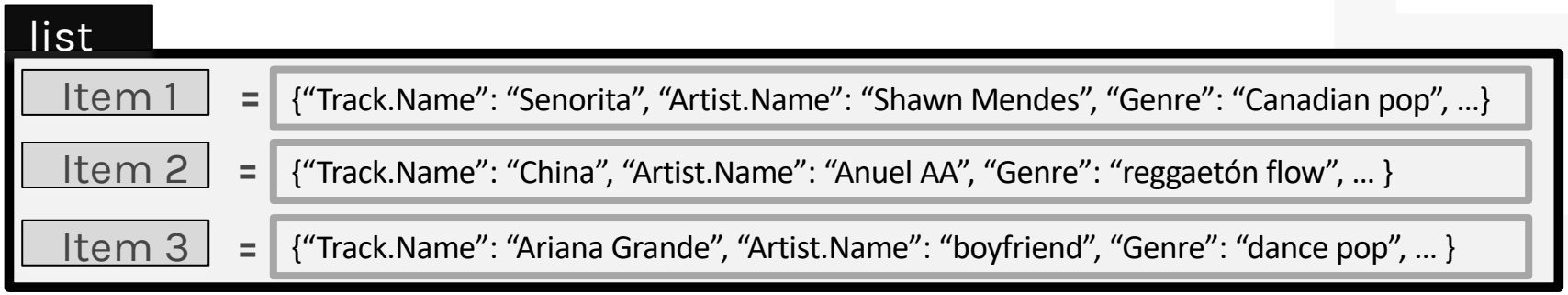
Possible Solution #2: A list of dictionaries

- Q4: Write code to print the songs (and their attributes), if we sorted by their popularity (highest scoring ones first).

EDA: LIST OF DICTIONARIES

Possible Solution #2: A list of dictionaries

- **Q4:** Write code to print the songs (and their attributes), if we sorted by their popularity (highest scoring ones first).

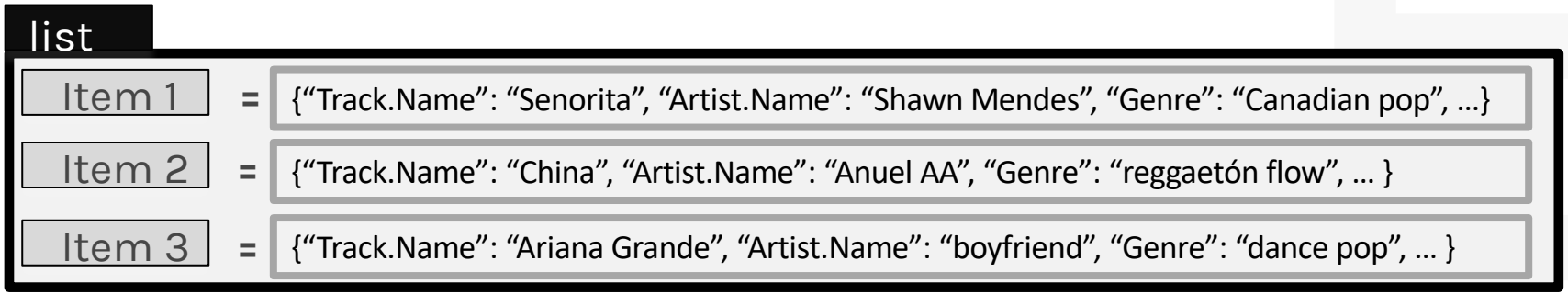


Cumbersome to move dictionaries around in a list. Problematic even if we don't move the dictionaries.

EDA: LIST OF DICTIONARIES

Possible Solution #2: A list of dictionaries

- **Q5:** How could you check for null/empty entries? This is only 50 entries. Imagine if we had 500,000.



EDA: LIST OF DICTIONARIES

Possible Solution #2: A list of dictionaries

- Q6: Imagine we had another table* below (i.e., .csv file). How could we combine its data with our already-existing *dataset*?

spotify_aux.csv

	TrackName	ArtistName	ExplicitLanguage	
1	Senorita	Shawn Menc	TRUE	
2	China	Anuel AA	FALSE	
3	boyfriend (w	Ariana Granc	TRUE	
4	Beautiful Pec	Ed Sheeran	FALSE	

* 3rd column is made-up by me. Random values. Pretend they're accurate.

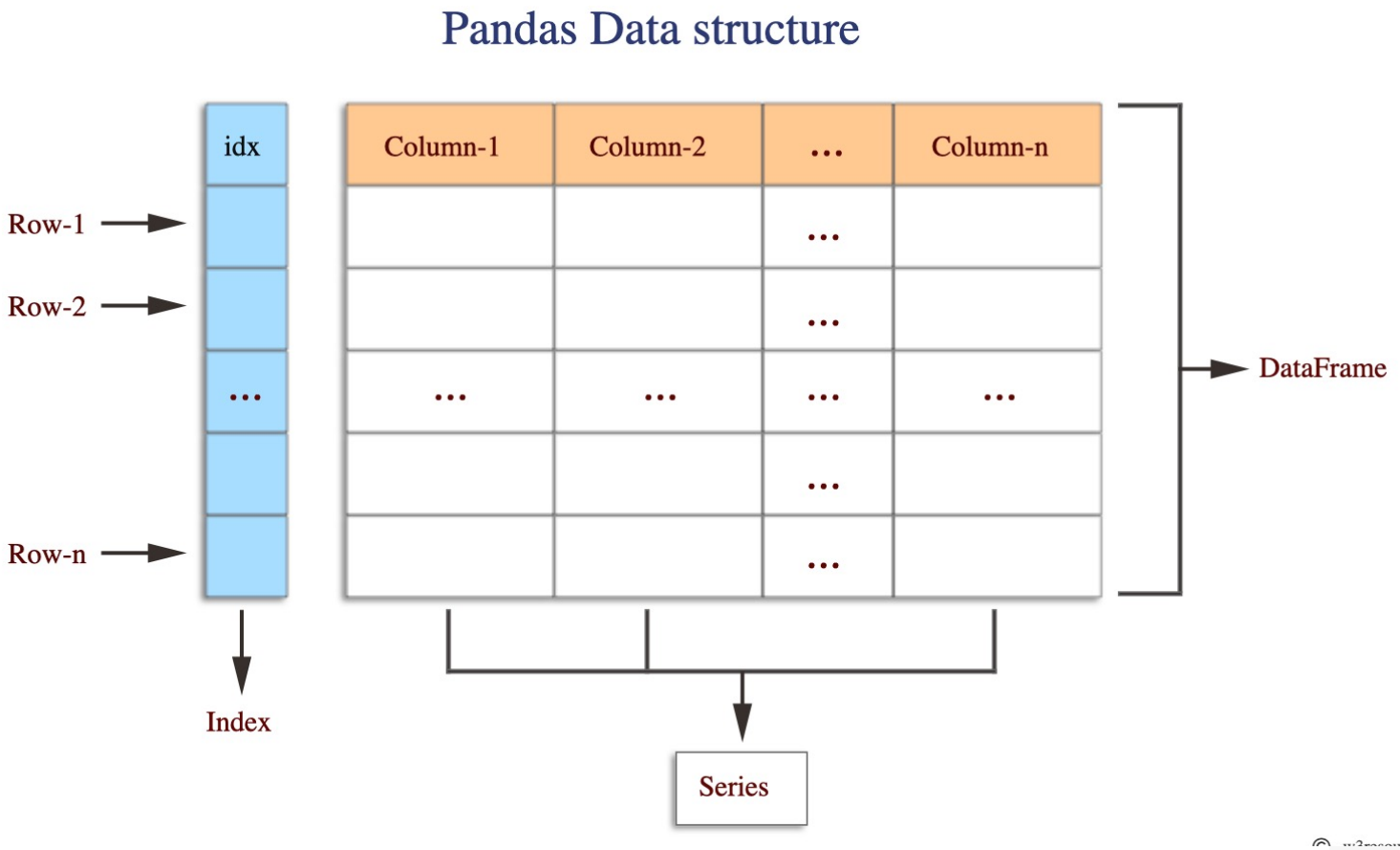
EDA: WITH PANDAS!

Kung Fu Panda is property of DreamWorks and Paramount Pictures

LECTURE OUTLINE

- Exploratory Data Analysis (EDA):
  - Without Pandas (part 1) – These slides
  - With Pandas (part 2) – Mostly Jupyter Notebook
- Data concerns (part 3) – These slides

PANDAS


Pandas Data structure
idx
Column-1
Column-2
...
Column-n
Row-1
Row-2
Row-n
DataFrame
Index
Series

PANDAS

Pandas Basic commands:

Imports the following commands to start:

```
import pandas as pd
import numpy as np
```

Pandas version:

```
import pandas as pd
print(pd.__version__)
```

HOW TO CREATE A SERIES FROM A LIST, NUMPY ARRAY AND DICT?

```
# Input
import numpy as np
a_list = list("abcdefg")
numpy_array = np.arange(1, 10)
dictionary = {"A":  0, "B":1, "C":2, "D":3, "E":5}
```

```
series1 = pd.Series(a_list)
print(series1)
series2 = pd.Series(numpy_array)
print(series2)
series3 = pd.Series(dictionary)
print(series3)
```

```
0    a
1    b
2    c
3    d
4    e
5    f
6    g
dtype: object
0    1
1    2
2    3
3    4
4    5
5    6
6    7
7    8
8    9
dtype: int64
A    0
B    1
C    2
D    3
E    5
dtype: int64
```

HOW TO COMBINE MANY SERIES TO FORM A DATAFRAME?

```
# input
ser1 = pd.Series(list('abcedfghijklmnopqrstuvwxyz'))
ser2 = pd.Series(np.arange(26))
```

```
# using pandas DataFrame
ser_df = pd.DataFrame(ser1, ser2).reset_index()
ser_df.head()
```

	index	0
0	0	a
1	1	b
2	2	c
3	3	e
4	4	d

EDA: WITH PANDAS

What / Why?

- Pandas is an *open-source* Python library (anyone can contribute)
- Allows for high-performance, easy-to-use data structures and data analysis
- Unlike NumPy library which provides multi-dimensional arrays, Pandas provides 2D table object called **DataFrame** (akin to a spreadsheet with column names and row labels).
- Used by *a lot* of people

EDA: WITH PANDAS

How

- import **pandas** library (convenient to rename it)
- Use **read_csv()** function

```
import pandas as pd
dataframe = pd.read_csv("yourfile.csv")
```

EDA: WITH PANDAS

Common Panda functions

- **High-level viewing:**
- head() – first N observations
- tail() – last N observations
- columns() – names of the columns
- describe() – statistics of the quantitative data
- dtypes() – the data types of the columns

EDA: WITH PANDAS

Common Panda functions

- **Accessing/processing:**
- df[“column_name”]
- Df.column_name
- .max(), .min(), .idxmax(), .idxmin()
- <dataframe> <conditional statement>
- .loc[] – label-based accessing
- .iloc[] – index-based accessing
- .sort_values()
- .isnull(), .notnull()

EDA: WITH PANDAS

Common Panda functions

- **Grouping/Splitting/Aggregating:**
  - groupby(), .get_groups()
  - .merge()
  - .concat()
  - .aggegate()
  - .append()

PANDAS EXAMPLE OUTLINES

- Import Dataset
- Display Dataset
- Merge Dataset
- Rebuild Dataset
- Dataset transformation
- Group by
- Joint

IMPORT DATASET

```
#importing the dataset by reading the csv file
data = pd.read_csv(/content/Iris.csv)
```

```
#displaying the first five rows of dataset
data.head()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

DISPLAY FIRST FIVE ROWS OF DATASET

In [49]:
```
dfcwci.head()
```

Out[49]:

	id	last_name	first_name	middle_name	street_1	street_2	city	state	zip	amount	date	candidate_id
0	NaN	Agee	Steven	NaN	549 Laurel Branch Road	NaN	Floyd	VA	24091	500.0	2007-06-30	16
1	NaN	Ahrens	Don	NaN	4034 Rennellwood Way	NaN	Pleasanton	CA	94566	250.0	2007-05-16	16
2	NaN	Ahrens	Don	NaN	4034 Rennellwood Way	NaN	Pleasanton	CA	94566	50.0	2007-06-18	16
3	NaN	Ahrens	Don	NaN	4034 Rennellwood Way	NaN	Pleasanton	CA	94566	100.0	2007-06-21	16
4	NaN	Akin	Charles	NaN	10187 Sugar Creek Road	NaN	Bentonville	AR	72712	100.0	2007-06-16	16

In [22]:
```
del dfcwci['id']
dfcwci.head()
```

Out[22]:

	last_name	first_name	middle_name	street_1	street_2	city	state	zip	amount	date	candidate_id
0	Agee	Steven	NaN	549 Laurel Branch Road	NaN	Floyd	VA	24091	500.0	2007-06-30	16
1	Ahrens	Don	NaN	4034 Rennellwood Way	NaN	Pleasanton	CA	94566	250.0	2007-05-16	16
2	Ahrens	Don	NaN	4034 Rennellwood Way	NaN	Pleasanton	CA	94566	50.0	2007-06-18	16
3	Ahrens	Don	NaN	4034 Rennellwood Way	NaN	Pleasanton	CA	94566	100.0	2007-06-21	16
4	Akin	Charles	NaN	10187 Sugar Creek Road	NaN	Bentonville	AR	72712	100.0	2007-06-16	16

HOW TO GET USEFUL INFOS

```
# input
state = np.random.RandomState(100)
ser = pd.Series(state.normal(10, 5, 25))
```

```
# using pandas
ser.describe()
```

```
count    25.000000
mean     10.435437
std       4.253118
min       1.251173
25%       7.709865
50%      10.922593
75%      13.363604
max      18.094908
dtype: float64
```

REBUILD DATASET (1)

- To find and fill the missing data in the dataset we will use another function.
- **Using isnull() /isna() function:**

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	False	False	False	False	False	False
1	False	False	False	False	False	False
2	False	False	False	False	False	False
3	False	False	False	False	False	False
4	False	False	False	False	False	False

REBUILD DATASET (2)

- **Using isna(). sum()**

```
data.isna().sum()
```

```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

REBUILD DATASET (3)

- **De-Duplicate**
- De-Duplicate means remove all duplicate values

data.duplicated()

```
0      False
1      False
2      False
3      False
4      False
       ...
145    False
146    False
147    False
148    False
149    False
Length: 150, dtype: bool
```

REBUILD DATASET(4)

- **DataFrame.fillna()**
- Fill NA/NaN values using the specified method.

pandas.DataFrame.fillna

```
DataFrame.fillna(value=None, method=None, axis=None, inplace=False, limit=None,
downcast=None)                                                          [source]
```

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.fillna.html

REBUILD DATASET (5)

- If a dataset contains duplicate values it can be removed using the drop_duplicates() function.

pandas.DataFrame.drop_duplicates

```
DataFrame.drop_duplicates(subset=None, keep='first', inplace=False,
ignore_index=False)
```

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.drop_duplicates.html

GROUPBY DATASET(1)

- Contributors

```
dfcwci=pd.read_csv("../data/contributors_with_candidate_id.txt", sep="|")
dfcwci.head()
```

	id	last_name	first_name	middle_name	street_1	street_2	city	state	zip	amount	date	candidate_id
0	NaN	Agee	Steven	NaN	549 Laurel Branch Road	NaN	Floyd	VA	24091	500.0	2007-06-30	16
1	NaN	Ahrens	Don	NaN	4034 Rennellwood Way	NaN	Pleasanton	CA	94566	250.0	2007-05-16	16
2	NaN	Ahrens	Don	NaN	4034 Rennellwood Way	NaN	Pleasanton	CA	94566	50.0	2007-06-18	16
3	NaN	Ahrens	Don	NaN	4034 Rennellwood Way	NaN	Pleasanton	CA	94566	100.0	2007-06-21	16
4	NaN	Akin	Charles	NaN	10187 Sugar Creek Road	NaN	Bentonville	AR	72712	100.0	2007-06-16	16

GROUPBY DATASET(2)

- Groupby:
  - Splitting the data into groups based on some criteria
  - Applying a function to each group independently
  - Combining the results into a data structure

```
In [28]: dfcwci.groupby("state").sum()
```

Out[28]:

state	zip	amount	candidate_id
AK	2985459621	1210.00	111
AR	864790	14200.00	192
AZ	860011121	120.00	37
CA	14736360720	-5013.73	600
CO	2405477834	-5823.00	111
CT	68901376	2300.00	35
DC	800341853	-1549.91	102
FL	8970626520	-4050.00	803

MERGE DATASET

- Merging the dataset is the process of combining two datasets in one.

pandas.DataFrame.merge ¶

```
DataFrame.merge(right, how='inner', on=None, left_on=None, right_on=None,
left_index=False, right_index=False, sort=False, suffixes=('_x', '_y'), copy=True,
indicator=False, validate=None)                                          [source]
```

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.merge.html

MERGE DATASET(1)

- Example: Candidates

```
In [8]: dfcand=pd.read_csv("../data/candidates.txt", sep='|')
        dfcand.head(10)
```

Out[8]:

	id	first_name	last_name	middle_name	party
0	33	Joseph	Biden	NaN	D
1	36	Samuel	Brownback	NaN	R
2	34	Hillary	Clinton	R.	D
3	39	Christopher	Dodd	J.	D
4	26	John	Edwards	NaN	D
5	22	Rudolph	Giuliani	NaN	R
6	24	Mike	Gravel	NaN	D
7	16	Mike	Huckabee	NaN	R
8	30	Duncan	Hunter	NaN	R
9	31	Dennis	Kucinich	NaN	D

```
dfcwci=pd.read_csv("../data/contributors_with_candidate_id.txt", sep="|")
dfcwci.head()
```

	id	last_name	first_name	middle_name	street_1	street_2	city	state	zip	amount	date	candidate_id
0	NaN	Agee	Steven	NaN	549 Laurel Branch Road	NaN	Floyd	VA	24091	500.0	2007-06-30	16
1	NaN	Ahrens	Don	NaN	4034 Rennellwood Way	NaN	Pleasanton	CA	94566	250.0	2007-05-16	16
2	NaN	Ahrens	Don	NaN	4034 Rennellwood Way	NaN	Pleasanton	CA	94566	50.0	2007-06-18	16
3	NaN	Ahrens	Don	NaN	4034 Rennellwood Way	NaN	Pleasanton	CA	94566	100.0	2007-06-21	16
4	NaN	Akin	Charles	NaN	10187 Sugar Creek Road	NaN	Bentonville	AR	72712	100.0	2007-06-16	16

MERGE DATASET(2)

- Merge:
- Combine tables on a common key-value

```
In [40]: cols_wanted=['last_name_x', 'first_name_x', 'candidate_id', 'id', 'last_name_y']
         dfcwci.merge(dfcand, left_on="candidate_id", right_on="id")[cols_wanted]
```

Out[40]:

	last_name_x	first_name_x	candidate_id	id	last_name_y
0	Agee	Steven	16	16	Huckabee
1	Akin	Charles	16	16	Huckabee
2	Akin	Mike	16	16	Huckabee
3	Akin	Rebecca	16	16	Huckabee
4	Aldridge	Brittni	16	16	Huckabee

MERGE DATASET: CONCERNS

- For example, say we have two datasets:

Dataset 1

Top 200 most-frequent streams per day (for June 2019)

SpotifySongID,	# of Streams,	Date
2789179,	42003,	06-01
•		
•		
3819390,	89103,	06-01
4492014,	52923,	06-02
•		
•		
8593013,	189145,	06-02

200

200

6,000 x 3

Dataset 2

Top 50 most streamed in 2019, so far

SpotifySongID,	Artist,	Track, [10 acoustic features]
2789179,	Billie Eilish,	bad guy, 3.2, 5.9, …
•		
•		
3901829,	Outkast,	Elevators, 9.3, 5,1, …

50

50 x 13

MERGE DATASET: CONCERNS

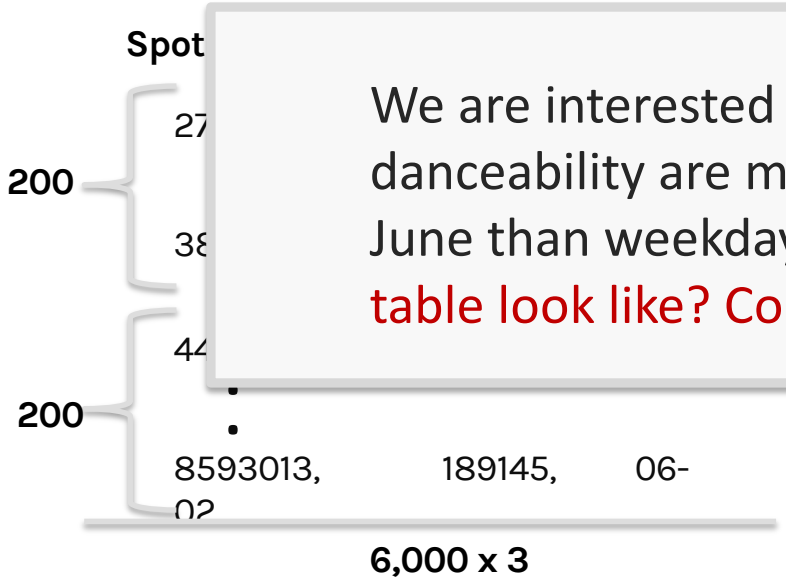
- For example, say we have two datasets:

Dataset 1

Top 200 most-frequent streams per day (for June 2019)

Dataset 2

Top 50 most streamed in 2019, so far



We are interested in determining if songs with high danceability are more popular during the weekends of June than weekdays in June. What should our merged table look like? Concerns?

MERGE DATASET: CONCERNS

- This is wasteful, as it has 10 acoustic features, artist, and track repeated many times for each unique song.

Datasets Merged (poorly)

SpotifySongID, # of Streams, Date, Artist, Track, [10 acoustic features]

200

2789179, 42003, 06-01 Billie Eilish, bad guy, 3.2, 5.9, ...

•
•

3819390, 89103, 06-01 Outkast, Elevators, 9.3, 5,1, ...

200

4492014, 52923, 06-02

•
•

8593013, 189145, 06-02

6,000 x 15 ➔ 90,000 cells

MERGE DATASET: CONCERNS

- Some rows may have null values for # of Streams (if the song wasn't popular in June)

Datasets Merged (better)

SpotifySongID, Artist, Track, [10 acoustic features], 06-01 Streams, 06-02 Streams

50

2789179, Billie Eilish, bad guy, 3.2, 5.9, ..., 42003, 42831, 43919

⋮

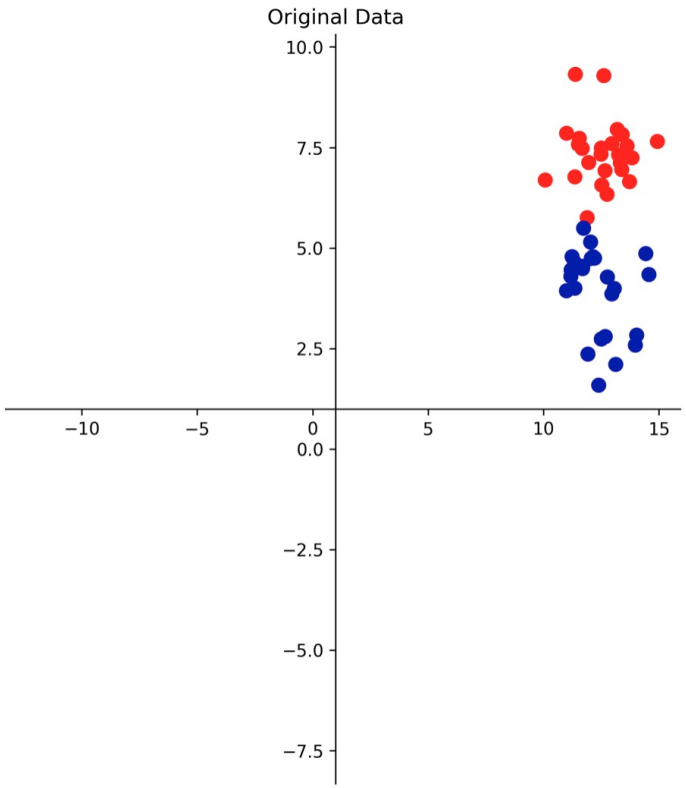
3901829, Outkast, Elevators, 9.3, 5,1, ... 29109, 27193, 25982

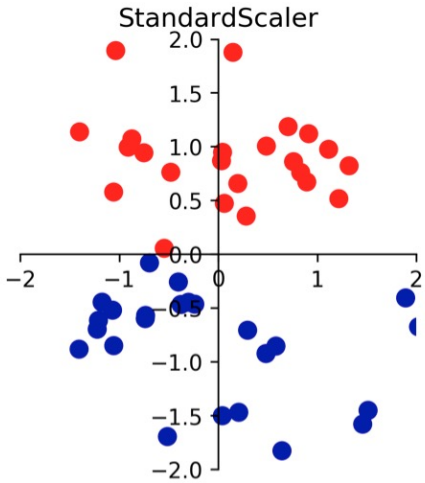
50 x 70 ➔ 3,500 cells

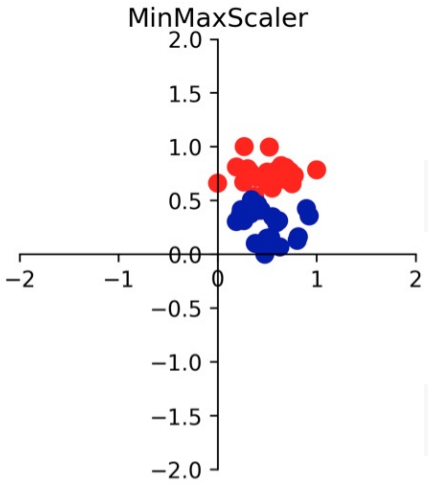
LECTURE OUTLINE

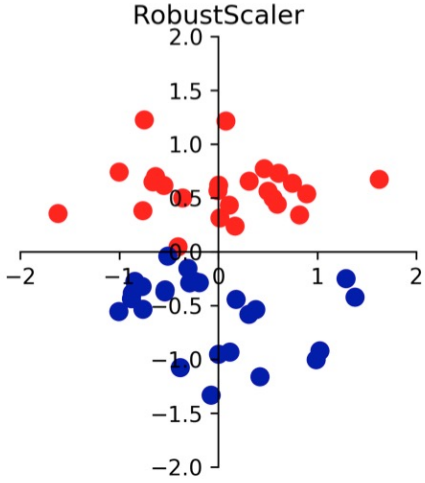
- Exploratory Data Analysis (EDA):
  - Without Pandas (part 1) – These slides
  - With Pandas (part 2) – Mostly Jupyter Notebook
- Data concerns (part 3) – These slides

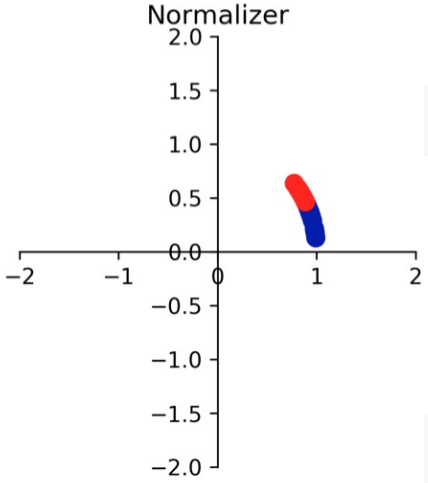
DATA CONCERNS: STANDARDIZATION AND NORMALIZATION

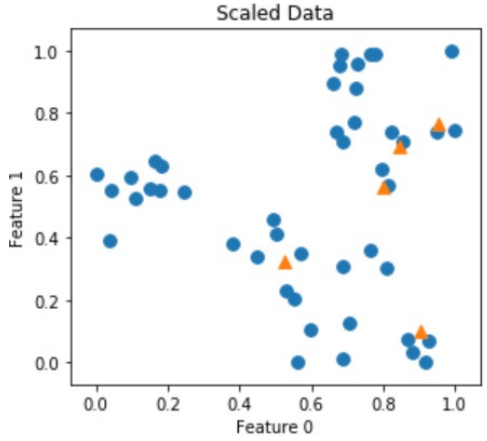


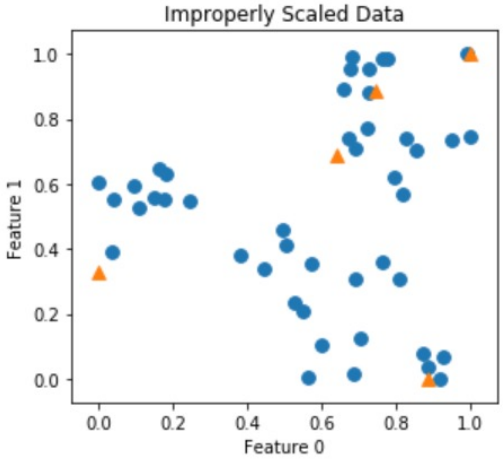












QUESTION

Case A:

1. Import Data
2. Split Data into training and test sets
3. Scale the training and test sets together using MinMaxScaler.
4. Visualization

Case B:

1. Import Data
2. Split Data into training and test sets
3. Scale training set using MinMaxScaler.
4. Rescale the test set separately using MinMaxScaler.
5. Visualization

QUESTION ---MAPPING

Following is a preview of the DataFrame `df`:

x	y	z
1	NaN	1
2	NaN	2
NaN	1	3

Match the commands to their expected outputs:

1. `df.notna().sum()`
2. `df.isna().any()`
3. `df['z'].isna()`

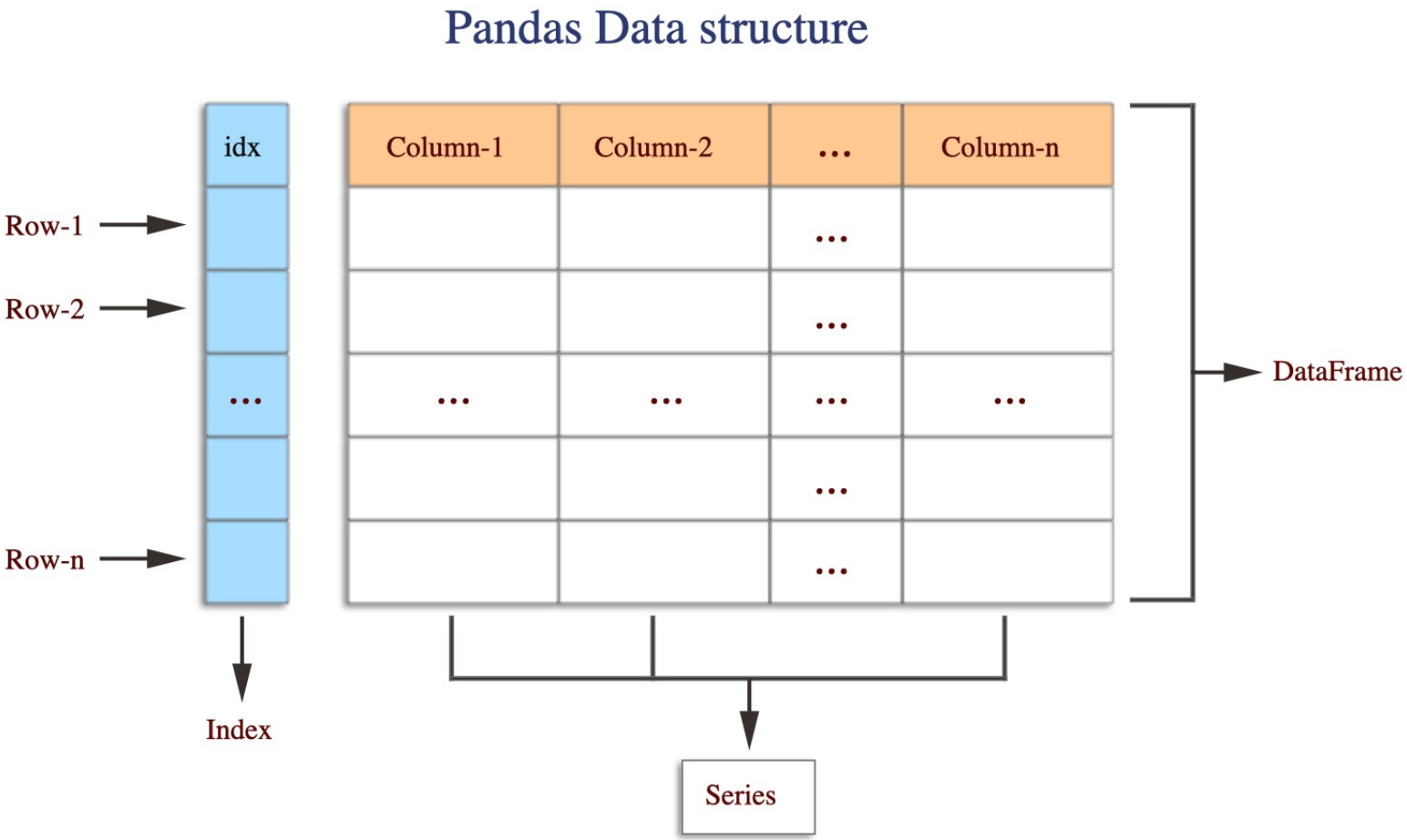
```
# A
0    False
1    False
2    False
Name: z, dtype: bool

# B
x    2
y    1
z    3
dtype: int64

# C
x     True
y     True
z    False
dtype: bool
```

QUESTION --- FILL IN A BLANK

- How to get useful infos of a Dataframe SER

- SER. describe()

QUESTION


Pandas Data structure
idx
Column-1
Column-2
...
Column-n
Row-1
Row-2
Row-n
DataFrame
Index
Series

GRAMMAR OF DATA

- If you need to find a Data Related work!!!

- https://www.w3resource.com/python-exercises/pandas/index.php

THANK YOU

VISIT US

WWW.XJTLU.EDU.CN

FOLLOW US

@XJTLU

Xi'an Jiaotong-Liverpool University
西交利物浦大學