Course Title:

Data Science (数据科学)

(Semester: Fall 2021)

Dr. Oluwarotimi W. SAMUEL

Research Center for Neural Engineering
Shenzhen Institutes of Advanced Technology
Chinese Academy of Sciences

Contact: (Email: samuel@siat.ac.cn & timitex92@gmail.com)

Phone: +86-15814491870

(2020.10.21)

Data Exploration and Cleaning

❑ Outline for today's lecture

- ✓ Data Exploration and Cleaning
- ✓ Practical Application using "A Case Study"
- ✓ Discussion of the Case Study.

Data Exploration and Cleaning

- **Objective:** This lecture will focus on Exploring and Cleaning Data with practical application using a real-life Case Study

- **Expectation:** At the end of this lecture, students are expected to understand how to explore and clean in the context of a Data Science project.

Data Exploration and Cleaning

❑ Overview Data Cleaning Process

Data cleaning/cleansing is the process of detecting and correcting corrupt or inaccurate records from a record set, table, or database.

It also refers to identifying incomplete, incorrect, inaccurate or irrelevant parts of the data and then replacing, modifying, or deleting the dirty or coarse data.

❑ Ov


Data Deduplication
Quality Check
Data Analysis
Data Cleansing
Data Normalization
Data Standardization

Data Cleaning

❑ Data Cleaning Process

- ✓ Data cleaning can be time consuming
- ✓ However, it is a critical stage of the Data Science lifecycle.

Data Cleaning

❑ Data Cleaning Process

Data cleaning often addresses:

- ✓ Missing values
- ✓ Formatting of values
- ✓ The overall structure of the data
- ✓ Extracting information from Complex values
- ✓ Unit conversion
- ✓ Interpretation of magnitudes, etc.

Data Cleaning

❑ Data Cleaning Process

Other Data cleaning examples:

- ✓ Misspellings
- ✓ Duplicated rows
- ✓ Inconsistent formats
- ✓ Unspecified units

Data Cleaning

❑Data Cleaning Process

We shall take a practical look at what a Data Cleaning Process looks like in the next few slides…

Data Cleaning Case Study

- **Case Study:** Using Data Recorded for Crime Incidence in Boston, USA, by the Boston Police Department from June 15, 2015 to date

https://data.boston.gov/dataset/crime-incident-reports-august-2015-to-date-source-new-system

Data Cleaning Case Study

❑ Case Study:

ANALYZE BOSTON

DATASETS NEWS TIPS LOG IN SIGN UP CONTACT

Home > Organizations > Boston Police Department > Crime Incident Reports ... > Crime Incident Reports ...

CRIME INCIDENT REPORTS (AUGUST 2015 - TO DATE) ...

DOWNLOAD DATA API

URL: https://data.boston.gov/dataset/6220d948-eae2-4e4b-8...

FROM THE DATASET ABSTRACT

Crime incident reports are provided by Boston Police Department (BPD) to document the initial details surrounding an incident to which BPD officers respond. This is a dataset containing...

Source: Crime Incident Reports (August 2015 - To Date) (Source: New System)

Data Table

Embed

Add Filter

Show 10 entries

Hide/Unhide Columns Copy Download Print

Showing 31 to 40 of 533,452 entries

Search:

_id	INCIDENT_NUMBER	OFFENSE_CODE	OFFENSE_CODE_GROUP	OFFENSE_DESCRIPTION	DISTRICT	REPORTING_AREA	SHOOTING	OCCURRED_ON_DATE	YEAR	MONTH	DAY_O
31	I20200718	3201	None	PROPERTY - LOST/ MISSING	E13	574	0	2020-01-25 12:00:00	2020	1	Saturda
32	I20200024	3803	None	M/V ACCIDENT - PERSONAL INJURY	D14		0	2020-01-02 07:00:00	2020	1	Thursda
33	I20000994	3005	None	SICK ASSIST	A7	28	0	2020-01-04 17:50:00	2020	1	Saturda
34	I19210875	3301	None	VERBAL DISPUTE	E13	912	0	2019-12-30 22:04:00	2019	12	Monday
35	I19210066	2647	None	THREATS TO DO BODILY HARM	C11	823	0	2019-12-14 15:00:00	2019	12	Saturda

https://data.boston.gov/dataset/crime-incident-reports-august-2015-to-date-source-new-system

Data Cleaning Case Study

Case Study: The Boston Police Department Provides Data of Recorded Crime Incidence from June 15, 2015 to date.

The data contains information such as:

- ✓ Type of incidence/crime
- ✓ The place of occurrence
- ✓ Time and date of occurrence
- ✓ Description of the crime

Data Cleaning Case Study

Case Study: The Boston Police Department Provides Data of Recorded Crime Incidence from June 15, 2015 to date.

Data Exploration may provide:

- ✓ What are the major crimes committed/year?
- ✓ What are the major crimes per year?
- ✓ What time do the crime occur?
- ✓ Which neighborhood has the most crime?
- ✓ Which are the safest neighborhood?

Data Cleaning Case Study

- **Case Study:** ***Boston Crime Data***

✓ Recall that the dataset for this Case Study can be accessed via **Direct download** or **Query via API**

✓ We shall proceed to explore the data and perform some data cleaning operations in the subsequent slides.

Data Cleaning Case Study

❑ Case Study:

✓ Recall we setup the platform for our practical by installing **Anaconda**, **PyCham**, **Package Installer**, and some **basic libraries/packages**.

✓ We shall now ***import*** some libraries needed to work with the ***Crime Data***.

pandas

numpy

matplotlib

seaborn

sklearn

Data Cleaning Case Study

❑ Case Study:

Before we begin coding, these are some important things to note

- ✓ Read the error message
- ✓ Search the error message on the internet
- ✓ Print things out a lot
- ✓ Run code after each change
- ✓ Make sure the coloring in your text editor looks like you expect it to.
- ✓ Comment your code
- ✓ Read the Python documentation for the methods you are employing.

Data Cleaning Case Study

❑ Case Study:

Import libraries: Now, **import** relevant libraries/packages required to work with the ***Crime Data***.

Let's now explore the data...

```
# -*- Data Science (Data Cleaning) -*- #

# Import relevant libraries
import pandas as pd # Pandas (pd) offers data structures and operations for manipulating numerical data
import numpy as np # Numpy(np) supports operations on large multi-dimensional arrays and matrices
import matplotlib.pyplot as plt #Matplotlib supports plotting data in Python
import seaborn as sn #Seaborn is a Python data visualization library based on matplotlib
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score # Machine learning library for the Python programming language
```

Data Cleaning Case Study

❑ Case Study:

Load the Data: Next, we read the ***CSV file*** holding the data using "**read_csv**" function provided by the Pandas library.

```
12
13    # Read the CSV file that contains the Crime data using "read_csv" function
14    crime = pd.read_csv('crime_boston.csv', low_memory=False)
15
```

If low_memory=False , **then whole columns will be read in first, and then the proper types determined**. For example, the column will be kept as objects (strings) as needed to preserve information. If low_memory=True (the default), then pandas reads in the data in chunks of rows, then appends them together. Jun 17, 2014

Data Cleaning Case Study

❑ Case Study:

Check the datatypes of the features: It maybe necessary to know the datatype for each feature in the dataset. Used "**.info()**" function to access such.

```
15
16  #Checking the datatypes of the features contained in the Crime data
17  crime.info()
18
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 533452 entries, 0 to 533451
Data columns (total 17 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   INCIDENT_NUMBER      533452 non-null  object
 1   OFFENSE_CODE         533452 non-null  int64
 2   OFFENSE_CODE_GROUP   426839 non-null  object
 3   OFFENSE_DESCRIPTION  533452 non-null  object
 4   DISTRICT             530984 non-null  object
 5   REPORTING_AREA       533452 non-null  object
 6   SHOOTING             108359 non-null  object
 7   OCCURRED_ON_DATE     533452 non-null  object
 8   YEAR                 533452 non-null  int64
 9   MONTH                533452 non-null  int64
 10  DAY_OF_WEEK          533452 non-null  object
 11  HOUR                 533452 non-null  int64
 12  UCR_PART             426729 non-null  object
 13  STREET               511660 non-null  object
 14  Lat                  503626 non-null  float64
 15  Long                 503626 non-null  float64
 16  Location             533452 non-null  object
dtypes: float64(2), int64(4), object(11)
memory usage: 69.2+ MB
```

Data Cleaning Case Study

❑ Case Study:

View the Data: Below is what the Crime report Dataframe looks like using the following command for display:

```
15
16    # Display the loaded Crime data to see what it looks like
17    crime  crime: {DataFrame: (533452, 17)}
```

	INCIDENT_NUMBER	OFFENSE_CODE	OFFENSE_CODE_GROUP	OFFENSE_DESCRIPTION	DISTRICT	REPORTING_AREA	SHOOTING	OCCURRED_ON_DATE	YEAR
0	TESTTEST2	423	NaN	ASSAULT - AGGRAVATED	External		0	2019-10-16 00:00:00	2019
1	S97333701	3301	NaN	VERBAL DISPUTE	C6	915	0	2020-07-18 14:34:00	2020
2	S47513131	2647	NaN	THREATS TO DO BODILY HARM	E18	530	0	2020-06-24 10:15:00	2020
3	I92102201	3301	NaN	VERBAL DISPUTE	E13	583	0	2019-12-20 03:08:00	2019
4	I92097173	3115	NaN	INVESTIGATE PERSON	C11	355	0	2019-10-23 00:00:00	2019
...	...	...	...	...	...	...	...	...	...
533447	020062356	1107	NaN	FRAUD - IMPERSONATION	E18	520	0	2020-08-28 18:39:00	2020
533448	020054040	3501	NaN	MISSING PERSON	C11		0	2020-07-30 15:30:00	2020

Data Cleaning Case Study

❑ Case Study:

Explore Data: Next, check if there exist ***missing values*** in the DataFrame using the "isnull" function.

```
18
19    # Check if there are missing values in the DataFrame using the method  "isnull()" function
20    crime.isnull()  crime: {DataFrame: (533452, 17)}
21
```

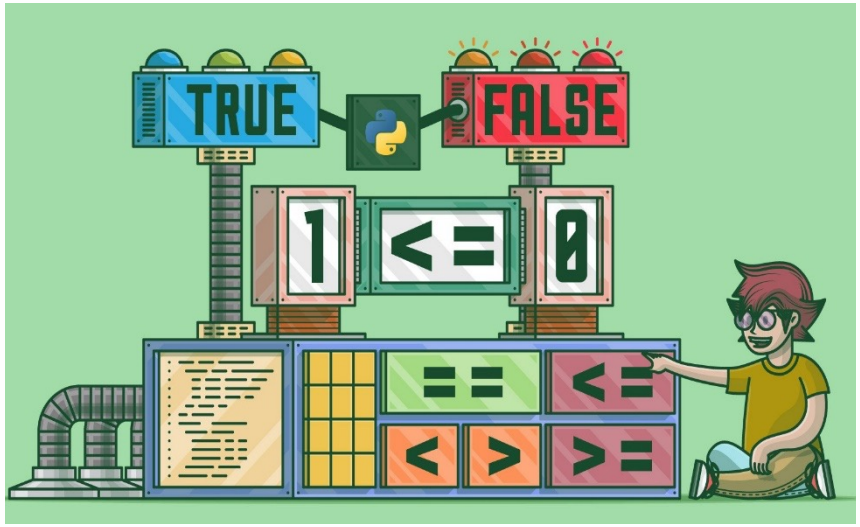
	INCIDENT_NUMBER	OFFENSE_CODE	OFFENSE_CODE_GROUP	OFFENSE_DESCRIPTION	DISTRICT	REPORTING_AREA	SHOOTING	OCCURRED_ON_DATE	YEAR
0	False	False	True	False	False	False	False	False	False
1	False	False	True	False	False	False	False	False	False
2	False	False	True	False	False	False	False	False	False
3	False	False	True	False	False	False	False	False	False
4	False	False	True	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...
533447	False	False	True	False	False	False	False	False	False
533448	False	False	True	False	False	False	False	False	False
533449	False	False	True	False	False	False	False	False	False
533450	False	False	True	False	False	False	False	False	False
533451	False	False	True	False	False	False	False	False	False

533452 rows × 17 columns

Data Cleaning Case Study

❑ Case Study:

Let's try to understand the outcome of the **"isnull()"** function which returns a Boolean value.



- ✓ Booleans are ***truth values*** in a computer programming language.
- ✓ "**False**" means the corresponding values in the crime database are not missing
- ✓ "**True**" means the corresponding values in the crime database are missing.

Data Cleaning Case Study

❑ Case Study:

As you can see a DataFrame is returned that contains Boolean values. Cells that contains "**True**" means that missing values exist in such locations.

	INCIDENT_NUMBER	OFFENSE_CODE	OFFENSE_CODE_GROUP	OFFENSE_DESCRIPTION	DISTRICT	REPORTING_AREA	SHOOTING	OCCURRED_ON_DATE	YEAR
0	False	False	True	False	False	False	False	False	False
1	False	False	True	False	False	False	False	False	False
2	False	False	True	False	False	False	False	False	False
3	False	False	True	False	False	False	False	False	False
4	False	False	True	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...
533447	False	False	True	False	False	False	False	False	False
533448	False	False	True	False	False	False	False	False	False
533449	False	False	True	False	False	False	False	False	False
533450	False	False	True	False	False	False	False	False	False
533451	False	False	True	False	False	False	False	False	False

533452 rows × 17 columns

Data Cleaning Case Study

❑ Case Study:

Only interested in series with missing values: What if we want access only series (column) that contains missing values in the crime database?

```
# To Access only the rows that contain missing values
crime.isnull().any(axis=1)  crime: {DataFrame: (533452, 17)}
```

```
0         True
1         True
2         True
3         True
4         True
          ...
533447    True
533448    True
533449    True
533450    True
533451    True
Length: 533452, dtype: bool
```

pandas.DataFrame.any

`DataFrame.any(axis=0, bool_only=None, skipna=True, level=None, **kwargs)` [source]

Return whether any element is True, potentially over an axis.

Returns False unless there is at least one element within a series or along a Dataframe axis that is True or equivalent (e.g. non-zero or non-empty).

Parameters: **axis** : *{0 or 'index', 1 or 'columns', None), default 0*

Indicate which axis or axes should be reduced.

- 0 / 'index' : reduce the index, return a Series whose index is the original column labels.
- 1 / 'columns' : reduce the columns, return a Series whose index is the original index.
- None : reduce all axes, return a scalar.

Data Cleaning Case Study

❑ Case Study:

Only interested in series with missing values: What if we want access only series (column) that contains missing values in the crime database?

```
21
22    # To Access only the series (columns) that contain missing values
23    crime.isnull().any(axis=0)   crime: {DataFrame: (533452, 17)}
24
```

```
21  INCIDENT_NUMBER          False
    OFFENSE_CODE             False
    OFFENSE_CODE_GROUP        True
    OFFENSE_DESCRIPTION      False
    DISTRICT                  True
    REPORTING_AREA           False
    SHOOTING                  True
    OCCURRED_ON_DATE         False
    YEAR                     False
    MONTH                    False
    DAY_OF_WEEK              False
    HOUR                     False
    UCR_PART                  True
    STREET                    True
    Lat                       True
    Long                      True
    Location                 False
    dtype: bool
```

pandas.DataFrame.any

`DataFrame.any(axis=0, bool_only=None, skipna=True, level=None, **kwargs)` [source]

Return whether any element is True, potentially over an axis.

Returns False unless there is at least one element within a series or along a Dataframe axis that is True or equivalent (e.g. non-zero or non-empty).

Parameters: **axis** : *{0 or 'index', 1 or 'columns', None}, default 0*

Indicate which axis or axes should be reduced.

- 0 / 'index' : reduce the index, return a Series whose index is the original column labels.
- 1 / 'columns' : reduce the columns, return a Series whose index is the original index.
- None : reduce all axes, return a scalar.

Data Cleaning Case Study

❑ Case Study:

If we are interested in accessing the rows with "NaN", then the following steps should be taken.

```
27
28  # Using the above series to obtain the rows with missing values
29  # And saving the resulting output into the variable "crime"
30  rows_with_missing_vals = crime.isnull().any(axis = 1)  crime: {DataFrame: (533452, 17)}
31  crime[rows_with_missing_vals]  crime: {DataFrame: (533452, 17)}  rows_with_missing_vals: {Series: (533452,)}
32
```

	INCIDENT_NUMBER	OFFENSE_CODE	OFFENSE_CODE_GROUP	OFFENSE_DESCRIPTION	DISTRICT	REPORTING_AREA	SHOOTING	OCCURRED_ON_DATE	YEAR
0	TESTTEST2	423	NaN	ASSAULT - AGGRAVATED	External		0	2019-10-16 00:00:00	2019
1	S97333701	3301	NaN	VERBAL DISPUTE	C6	915	0	2020-07-18 14:34:00	2020
2	S47513131	2647	NaN	THREATS TO DO BODILY HARM	E18	530	0	2020-06-24 10:15:00	2020
3	I92102201	3301	NaN	VERBAL DISPUTE	E13	583	0	2019-12-20 03:08:00	2019
4	I92097173	3115	NaN	INVESTIGATE PERSON	C11	355	0	2019-10-23 00:00:00	2019
...	...	...	...	...	...	...	...	...	...
533447	020062356	1107	NaN	FRAUD - IMPERSONATION	E18	520	0	2020-08-28 18:39:00	2020
533448	020054040	3501	NaN	MISSING PERSON	C11		0	2020-07-30 15:30:00	2020
533449	020046400	1501	NaN	WEAPON VIOLATION - CARRY/ POSSESSING/	B2	330	0	2020-07-02 01:38:00	2020

Data Cleaning Case Study

❑ Case Study:

Looking at the resulting data (**cleaned_crime**), it appears we need to drop the **year**, **month**, and **hours** **columns**, since they could be obtained from the date column. What about this?

```
32
33  # Clean the crime data by dropping the Year, Month, and Hour columns
34  # Since they occurred on date column which already includes date and time
35  cleaned_crime = crime.drop(columns = ['YEAR','MONTH', 'HOUR'])  crime: {DataFrame: (533452, 17)}
36  cleaned_crime  cleaned_crime: {DataFrame: (533452, 14)}
37
```

	INCIDENT_NUMBER	OFFENSE_CODE	OFFENSE_CODE_GROUP	OFFENSE_DESCRIPTION	DISTRICT	REPORTING_AREA	SHOOTING	OCCUR
0	TESTTEST2	423	NaN	ASSAULT - AGGRAVATED	External		0	2019-1(
1	S97333701	3301	NaN	VERBAL DISPUTE	C6	915	0	2020-07
2	S47513131	2647	NaN	THREATS TO DO BODILY HARM	E18	530	0	2020-0(
3	I92102201	3301	NaN	VERBAL DISPUTE	E13	583	0	2019-12
4	I92097173	3115	NaN	INVESTIGATE PERSON	C11	355	0	2019-1(
...	...	...	...	...	...	...	...	...
533447	020062356	1107	NaN	FRAUD - IMPERSONATION	E18	520	0	2020-0(
533448	020054040	3501	NaN	MISSING PERSON	C11		0	2020-07
533449	020046400	1501	NaN	WEAPON VIOLATION - CARRY/ POSSESSING/ SALE/ TR...	B2	330	0	2020-07
533450	020038446	1501	NaN	WEAPON VIOLATION - CARRY/ POSSESSING/ SALE/ TR...	B2	300	0	2020-0(
533451	020030892	540	NaN	BURGLARY - COMMERICAL	C11	380	0	2020-05

533452 rows × 14 columns

Data Cleaning Case Study

❑ Case Study:

Again, looking at the dataset closely, we have the following columns: '**Location**', '**Longitude**', and '**Latitude**'.

Meanwhile, the '**Location**' column contains exactly the information that are in the **Latitude** and **Longitude** columns and the **Location** column may not be relevant.

So, we may drop **Location** column to have a more compact dataset.

Data Cleaning Case Study

❑ Case Study:

```
# Because we have 'Location' column that contains the latitude and longitude geographic information
# And also the 'Latitude' and 'Longitude' columns, there may be no need for 'location' column.
# So, we shall drop it, to have a more compact data
cleaned_crime = cleaned_crime.drop(columns='Location')  cleaned_crime: {DataFrame: (533452, 13)}
cleaned_crime  cleaned_crime: {DataFrame: (533452, 13)}
```

22

	INCIDENT_NUMBER	OFFENSE_CODE	OFFENSE_CODE_GROUP	OFFENSE_DESCRIPTION	DISTRICT	REPORTING_AREA	SHOOTING	OCCURRED_ON_DATE	DAY_OF_WEEK	UCR_PART	ST
0	TESTTEST2	423	NaN	ASSAULT - AGGRAVATED	External		0	2019-10-16 00:00:00	Wednesday	NaN	RIVERVIE
1	S97333701	3301	NaN	VERBAL DISPUTE	C6	915	0	2020-07-18 14:34:00	Saturday	NaN	MARY BC WAY
2	S47513131	2647	NaN	THREATS TO DO BODILY HARM	E18	530	0	2020-06-24 10:15:00	Wednesday	NaN	READVILL
3	I92102201	3301	NaN	VERBAL DISPUTE	E13	583	0	2019-12-20 03:08:00	Friday	NaN	DAY ST
4	I92097173	3115	NaN	INVESTIGATE PERSON	C11	355	0	2019-10-23 00:00:00	Wednesday	NaN	GIBSON S
...	...	...	...	...	...	...	...	...	...	...	...
533447	020062356	1107	NaN	FRAUD - IMPERSONATION	E18	520	0	2020-08-28 18:39:00	Friday	NaN	HYDE PA AVE
533448	020054040	3501	NaN	MISSING PERSON	C11		0	2020-07-30 15:30:00	Thursday	NaN	GIBSON S
533449	020046400	1501	NaN	WEAPON VIOLATION - CARRY/ POSSESSING/ SALE/ TR...	B2	330	0	2020-07-02 01:38:00	Thursday	NaN	PASADEN RD
533450	020038446	1501	NaN	WEAPON VIOLATION - CARRY/ POSSESSING/ SALE/ TR...	B2	300	0	2020-06-03 01:15:00	Wednesday	NaN	WASHIN ST
533451	020030892	540	NaN	BURGLARY -	C11	380	0	2020-05-03 00:00:00	Sunday	NaN	GALLIVA

Data Cleaning Case Study

❑ Case Study:

Note: We shall use the "**cleaned_crime**" data going forward.

Next, let's check for **misspelling** on specific column/s. For example, let's do that on '**OFFENSE_CODE_GROUP**'.

Data Cleaning Case Study

```
54  # Checking for misspellings in the "OFFENSE_CODE_GROUP"
55  size_cleaned_crime = cleaned_crime['OFFENSE_CODE_GROUP'].unique()  cleaned_crime: {DataFrame: (533452, 13)}
56  print(size_cleaned_crime)  size_cleaned_crime: {ndarray: (68,)}
57  size_cleaned_crime.siz
58
```

❑ Case Study:

```
[nan 'Auto Theft' 'Investigate Property' 'Investigate Person' 'Vandalism'
 'Verbal Disputes' 'Motor Vehicle Accident Response' 'Aggravated Assault'
 'Residential Burglary' 'Larceny' 'Firearm Violations'
 'Medical Assistance' 'Simple Assault' 'Missing Person Reported' 'Robbery'
 'Property Lost' 'Violations' 'Warrant Arrests' 'Firearm Discovery'
 'Other' 'Ballistics' 'Towed' 'Drug Violation' 'Fire Related Reports'
 'Fraud' 'Disorderly Conduct' 'Larceny From Motor Vehicle'
 'Police Service Incidents' 'Missing Person Located' 'Harassment'
 'Property Found' 'Liquor Violation' 'Property Related Damage'
 'Confidence Games' 'Commercial Burglary' 'Recovered Stolen Property'
 'Homicide' 'Other Burglary' 'Assembly or Gathering Violations'
 'Counterfeiting' 'Prisoner Related Incidents'
 'License Plate Related Incidents' 'Restraining Order Violations'
 'Search Warrants' 'License Violation' 'Landlord/Tenant Disputes'
 'Auto Theft Recovery' 'Operating Under the Influence' 'Evading Fare'
 'Embezzlement' 'Criminal Harassment' 'Harbor Related Incidents' 'Service'
 'Offenses Against Child / Family' 'Prostitution' 'Biological Threat'
 'Explosives' 'Arson' 'Aircraft'
 'HUMAN TRAFFICKING - INVOLUNTARY SERVITUDE' 'HOME INVASION'
 'Phone Call Complaints' 'Bomb Hoax' 'Manslaughter' 'HUMAN TRAFFICKING'
 'Gambling' 'INVESTIGATE PERSON' 'Burglary - No Property Taken']
51  68
```

Data Cleaning Case Study

❑ Case Study:

Looking at the above results, it can be seen that there are no misspellings observed but a few "**OFFENSE_CODE_GROUP**" have different style such as:

- ✓ **HUMAN TRAFFICKING - INVOLUNTARY SERVITUDE**
- ✓ **HOME INVASION**
- ✓ **HUMAN TRAFFICKING**
- ✓ **INVESTIGATE PERSON**

And we may need to reformat them to be consistent with the others

Data Cleaning Case Study

❑ Case Study:

We can achieve this by using the replace function ".replace()" which allows us to replace certain string in a DataFrame with string/value of our choice.

Thus, we shall replace the above "OFFENSE_CODE_GROUP" with:

- ✓ **Human Trafficking - Involuntary Servitude**
- ✓ **Home Invasion**
- ✓ **Human Trafficking**
- ✓ **Investigate Person**

Data Cleaning Case Study

❑ Case Study:

```
46
47  #Ensuring that all the OFFENSE_CODE_GROUP are in a consistent format.
48  cleaned_crime['OFFENSE_CODE_GROUP'] = cleaned_crime['OFFENSE_CODE_GROUP'].replace({'HUMAN TRAFFICKING - INVOLUNTARY SERVITUDE':'Human Traffick
49  cleaned_crime['OFFENSE_CODE_GROUP'] = cleaned_crime['OFFENSE_CODE_GROUP'].replace({'HOME INVASION':' Home Invasion'})  cleaned_crime: {DataFr
50  cleaned_crime['OFFENSE_CODE_GROUP'] = cleaned_crime['OFFENSE_CODE_GROUP'].replace({'HUMAN TRAFFICKING':'Human Trafficking'})  cleaned_crime:
51  cleaned_crime['OFFENSE_CODE_GROUP'] = cleaned_crime['OFFENSE_CODE_GROUP'].replace({'INVESTIGATE PERSON':'Investigate Person'})  cleaned_crime
52  # cleaned_crime['OFFENSE_CODE_GROUP'] = cleaned_crime['OFFENSE_CODE_GROUP'].replace({'HUMAN TRAFFICKING - INVOLUNTARY SERVITUDE':'Human Traff
53
```

```
[nan 'Auto Theft' 'Investigate Property' 'Investigate Person' 'Vandalism'
 'Verbal Disputes' 'Motor Vehicle Accident Response' 'Aggravated Assault'
 'Residential Burglary' 'Larceny' 'Firearm Violations'
 'Medical Assistance' 'Simple Assault' 'Missing Person Reported' 'Robbery'
 'Property Lost' 'Violations' 'Warrant Arrests' 'Firearm Discovery'
 'Other' 'Ballistics' 'Towed' 'Drug Violation' 'Fire Related Reports'
 'Fraud' 'Disorderly Conduct' 'Larceny From Motor Vehicle'
 'Police Service Incidents' 'Missing Person Located' 'Harassment'
 'Property Found' 'Liquor Violation' 'Property Related Damage'
 'Confidence Games' 'Commercial Burglary' 'Recovered Stolen Property'
 'Homicide' 'Other Burglary' 'Assembly or Gathering Violations'
 'Counterfeiting' 'Prisoner Related Incidents'
 'License Plate Related Incidents' 'Restraining Order Violations'
 'Search Warrants' 'License Violation' 'Landlord/Tenant Disputes'
 'Auto Theft Recovery' 'Operating Under the Influence' 'Evading Fare'
 'Embezzlement' 'Criminal Harassment' 'Harbor Related Incidents' 'Service'
 'Offenses Against Child / Family' 'Prostitution' 'Biological Threat'
 'Explosives' 'Arson' 'Aircraft'
 'Human Trafficking - Involuntary Servitude' ' Home Invasion'
 'Phone Call Complaints' 'Bomb Hoax' 'Manslaughter' 'Human Trafficking'
 'Gambling' 'Burglary - No Property Taken']
67
```

Data Cleaning Case Study

❑ Case Study:

Next, let's check for **misspelling** on another column, "OFFENSE_DESCRIPTION".

If such exist, then we shall try to tackle. Also, we shall try to ensure that all "OFFENSE_DESCRIPTION" are in a consistent format as in the above example of "OFFENSE_CODE_GROUP"

CLASS---WORK....

Data Cleaning Case Study

❑ Case Study:

```
# Checking for misspellings in the "OFFENSE_DESCRIPTION"
cleaned_crime['OFFENSE_DESCRIPTION'].unique()  cleaned_crime: {DataFrame: (533452, 13)}
```

CLASS---WORK

```
19 array(['ASSAULT - AGGRAVATED', 'VERBAL DISPUTE',
          'THREATS TO DO BODILY HARM', 'INVESTIGATE PERSON',
          'WARRANT ARREST - OUTSIDE OF BOSTON WARRANT', 'SICK ASSIST',
          'VANDALISM', 'M/V - LEAVING SCENE - PROPERTY DAMAGE',
          'LARCENY ALL OTHERS', 'PROPERTY - LOST/ MISSING',
          'PROPERTY - FOUND', 'AUTO THEFT - MOTORCYCLE / SCOOTER',
          'LARCENY THEFT FROM BUILDING', 'HARASSMENT/ CRIMINAL HARASSMENT',
          'LARCENY SHOPLIFTING', 'M/V ACCIDENT - PROPERTY DAMAGE',
          'TOWED MOTOR VEHICLE', 'ASSAULT - SIMPLE',
          'VAL - OPERATING AFTER REV/SUSP.', 'SICK/INJURED/MEDICAL - POLICE',
          'M/V ACCIDENT - PERSONAL INJURY',
          'M/V ACCIDENT - OTHER CITY VEHICLE',
          'LARCENY THEFT OF MV PARTS & ACCESSORIES', 'AUTO THEFT',
          'AUTO THEFT - LEASED/RENTED VEHICLE', 'INVESTIGATE PROPERTY',
          'M/V ACCIDENT INVOLVING PEDESTRIAN - INJURY',
          'BURGLARY - RESIDENTIAL - ATTEMPT',
          'WEAPON - FIREARM - CARRYING / POSSESSING, ETC',
          'SICK/INJURED/MEDICAL - PERSON', 'ASSAULT SIMPLE - BATTERY',
          'MISSING PERSON', 'ROBBERY - STREET', 'PROPERTY - LOST',
          'VAL - OPERATING UNREG/UNINS CAR', 'WARRANT ARREST',
          'FIREARM/WEAPON - FOUND OR CONFISCATED',
          'WEAPON - OTHER - OTHER VIOLATION',
          'ASSAULT - AGGRAVATED - BATTERY', 'BALLISTICS EVIDENCE/FOUND',
          'BURGLARY - RESIDENTIAL - NO FORCE',
          'M/V ACCIDENT - POLICE VEHICLE',
          'DRUGS - POSS CLASS B - COCAINE, ETC.',
          'DRUGS - POSS CLASS B - INTENT TO MFR DIST DISP',
          'M/V ACCIDENT - OTHER', 'FIRE REPORT - HOUSE, BUILDING, ETC.',
```

CLASS---WORK

Data Cleaning Case Study

- **Case Study:**

Again, there are no misspellings observed in 'OFFENSE _DESCRIPTION' column.

In addition, we shall be checking for misspellings is the 'DAY_OF_WEEK'

Data Cleaning Case Study

❑ Case Study:

```
# Checking for misspellings in the "DAY_OF_WEEK"
cleaned_crime['DAY_OF_WEEK'].unique()  cleaned_crime: {DataFrame: (533452, 14)}
```

```
20  array(['Wednesday', 'Saturday', 'Friday', 'Tuesday', 'Thursday', 'Sunday',
        'Monday'], dtype=object)
```

Again, there are no misspellings observed for = ‘DAY_OF_WEEK’ column.

Data Cleaning Case Study

❑ Case Study:

- ✓ Cleaning every aspect of a dataset may take too long
- ✓ On the other hand, Not cleaning the dataset at all, will lead to drawing inaccurate conclusions.
- ✓ So, it is important to find a balance when cleaning a data
- ✓ The decision you make during data cleaning impact all the analysis you perform afterwards.

Data Cleaning Case Study

❑ Case Study:

This is the time to discuss what we have learnt during today's lecture especially the hands-on section...

❑ Summary for today's lecture

- ✓ We discussed about the need for Data Exploration and Cleaning in the context of Data Science.

- ✓ Further, we demonstrated the above concept using dataset of Crime Incidences Recorded by the Boston Police Department from 2015 to date.

Questions and Comments!