DSC540-Project Milestone3-Jyoti Dave

January 27, 2025

[1]:
```
# Weeks 7 & 8 Term project : Milestone 3
```

[2]:
```
# Cleaning/Formatting Website Data
```

[3]:
```
## Perform at least 5 data transformation and/or cleansing steps to your␣
 ↪website data.

# Examples:
# •         Replace Headers
# •         Format data into a more readable format
# •         Identify outliers and bad data
# •         Find duplicates
# •         Fix casing or inconsistent values
# •         Conduct Fuzzy Matching
```

[4]:
```
# Reading tabular data from a web page and creating dataframes
# from https://en.wikipedia.org/wiki/
 ↪Statistics_of_the_COVID-19_pandemic_in_the_United_States

import pandas as pd

#read the wikipedia page
list_of_df = pd.read_html("https://en.wikipedia.org/wiki/
 ↪Statistics_of_the_COVID-19_pandemic_in_the_United_States",header=0)
# total number of dataframe
len(list_of_df)
```

[4]:
```
72
```

[5]:
```
# Print the size dataframes to determine which one we are interested
for t in list_of_df:
    print(t.shape)
```

```
(0, 2)
(59, 7)
(0, 2)
(0, 2)
(0, 2)
```

```
(0, 2)
(0, 2)
(12, 3)
(0, 2)
(0, 2)
(0, 2)
(0, 2)
(0, 2)
(0, 2)
(0, 2)
(0, 2)
(0, 2)
(0, 2)
(192, 2)
(113, 2)
(6, 2)
(4, 2)
(1, 2)
(33, 2)
(31, 2)
(7, 2)
(1, 2)
(2, 2)
(5, 2)
(4, 2)
(1, 2)
(3, 2)
(35, 2)
(33, 2)
(1, 2)
(23, 2)
(5, 2)
(6, 2)
(6, 2)
(2, 2)
(3, 2)
(1, 2)
(9, 2)
(7, 2)
(1, 2)
(3, 2)
(8, 2)
(6, 2)
(1, 2)
(1, 2)
(22, 2)
(2, 2)
(5, 2)
```

```
(1, 2)
(1, 2)
(3, 2)
(1, 2)
(2, 2)
(36, 2)
(2, 2)
(0, 2)
(18, 2)
(17, 2)
(4, 2)
(3, 2)
(3, 2)
(1, 2)
(4, 2)
(3, 2)
(8, 2)
(7, 2)
(1, 2)
```

[6]:
```
# The second dataframe with index 1 is for "COVID-19 pandemic in the United␣
 ↪States by state and territory"
df=list_of_df[1]
df.head(5)
```

[6]:

	Unnamed: 0	Location[i]	Cases[ii]	Deaths[iii]	Recoveries[iv]	\
0	NaN	56 / 56	112168104	1168021	-	
1	NaN	Alabama	1659966	21138	509800	
2	NaN	Alaska	310531	1485	7165	
3	NaN	American Samoa	8326	34	3	
4	NaN	Arizona	2611788	34545	-	

	Hospital[v]	Ref.
0	-	NaN
1	50767	[11]
2	4208	[12]
3	-	[13]
4	157969	[14]

[7]:
```
# Step1: Replace Headers

# Column name doesn't seem to be correct in the above table. So replace it with␣
 ↪the meaning full names.
df.columns = ["Index", "Location", "Cases", "Deaths", "Recoveries",␣
 ↪"Hospitalizations", "Reference"]

# Print the updated headers
```

```
print("Updated Headers:")
df.head(5)
```

```
Updated Headers:
```

[7]:

	Index	Location	Cases	Deaths	Recoveries	Hospitalizations	\
0	NaN	56 / 56	112168104	1168021	-	-	
1	NaN	Alabama	1659966	21138	509800	50767	
2	NaN	Alaska	310531	1485	7165	4208	
3	NaN	American Samoa	8326	34	3	-	
4	NaN	Arizona	2611788	34545	-	157969	

	Reference
0	NaN
1	[11]
2	[12]
3	[13]
4	[14]

[8]:
```
# Rename headers for simplicity
# Step2: Format data into a more readable format
# First row shows the sub of data in each column which is not required here. Remove the first row
df = df.iloc[1:]
df.head()
```

[8]:

	Index	Location	Cases	Deaths	Recoveries	Hospitalizations	Reference
1	NaN	Alabama	1659966	21138	509800	50767	[11]
2	NaN	Alaska	310531	1485	7165	4208	[12]
3	NaN	American Samoa	8326	34	3	-	[13]
4	NaN	Arizona	2611788	34545	-	157969	[14]
5	NaN	Arkansas	1039712	13787	992651	48032	[15]

[9]:
```
# Step 3. Identify outliers and bad data
# Replace missing or invalid values (e.g., "-") with 0
df = df.replace(["-", "NaN"], 0)
# Remove the last column as it's not required
df = df.drop(df.columns[6], axis=1)
df.head()
```

[9]:

	Index	Location	Cases	Deaths	Recoveries	Hospitalizations
1	NaN	Alabama	1659966	21138	509800	50767
2	NaN	Alaska	310531	1485	7165	4208
3	NaN	American Samoa	8326	34	3	0
4	NaN	Arizona	2611788	34545	0	157969
5	NaN	Arkansas	1039712	13787	992651	48032

[10]:
```
# Step4: Remove unwanted column. Remove the first column as it's not required
df = df.drop(df.columns[0], axis=1)
```

[11]:
```
df.head(50)
```

[11]:

	Location	Cases	Deaths	Recoveries	\
1	Alabama	1659966	21138	509800	
2	Alaska	310531	1485	7165	
3	American Samoa	8326	34	3	
4	Arizona	2611788	34545	0	
5	Arkansas	1039712	13787	992651	
6	California	14332727	107703	0	
7	Colorado	1884386	16062	0	
8	Connecticut	983652	12354	0	
9	Delaware	351420	3682	18371	
10	District of Columbia	182395	1434	34985	
11	Florida[vi]	8063346	95592	0	
12	Georgia	3293182	44201	0	
13	Guam	64279	419	63816	
14	Hawaii	419655	2174	11958	
15	Idaho	526118	5766	92573	
16	Illinois	4139537	42033	0	
17	Indiana	2210538	28082	1881771	
18	Iowa	908936	10797	286309	
19	Kansas	946564	10229	0	
20	Kentucky	1808735	19914	53643	
21	Louisiana	1683744	19727	429935	
22	Maine	347116	3417	12975	
23	Maryland	1454101	17995	0	
24	Massachusetts	2374055	25822	644061	
25	Michigan	3313807	44966	1421905	
26	Minnesota	1903408	15990	1529440	
27	Mississippi	1000415	15480	774429	
28	Missouri	1790525	22931	0	
29	Montana	333758	3712	329725	
30	Nebraska	604901	5034	142336	
31	Nevada	924325	12508	0	
32	New Hampshire	382242	3340	378906	
33	New Jersey	3316021	36902	0	
34	New Mexico	727786	9236	660313	
35	New York	7975950	65835	475270	
36	North Carolina	3501404	29059	3371565	
37	North Dakota	310409	2233	236878	
38	Northern Mariana Islands	13981	46	13124	
39	Ohio	3747050	43958	3693448	
40	Oklahoma	1306350	16435	1288527	
41	Oregon	975856	10357	0	

42	Pennsylvania	3565644	53837	1843620	
43	Puerto Rico	1486077	7362	442126	
44	Rhode Island	470368	4365	0	
45	South Carolina	1859979	20353	559814	
46	South Dakota	305444	3401	275931	
47	Tennessee	2736444	30811	1996027	
48	Texas	9198592	94912	4,445,607[vii]	
49	US Virgin Islands	26148	133	26002	
50	Utah	1138594	5615	1103895	

	Hospitalizations
1	50767
2	4208
3	0
4	157969
5	48032
6	664057
7	109315
8	12257
9	36436
10	0
11	78472
12	149236
13	0
14	14887
15	19729
16	239809
17	194280
18	0
19	20081
20	78142
21	0
22	9316
23	52646
24	124678
25	0
26	96724
27	14042
28	0
29	14414
30	31570
31	0
32	9441
33	185627
34	40692
35	471317
36	194248

```
37             7831
38              311
39           151492
40            45990
41            41388
42                0
43                0
44            23606
45                0
46            14160
47            56696
48                0
49                0
50            43431
```

[12]:
```
# Step4:        Find duplicates
# Find duplicate rows
duplicates = df[df.duplicated(subset=["Location"], keep=False)]

print("Duplicate Rows:")
print(duplicates)

# Remove duplicate column if present (keep the first occurrence)
df = df.drop_duplicates(subset=["Location"], keep="first")
df.head()
```

```
Duplicate Rows:
Empty DataFrame
Columns: [Location, Cases, Deaths, Recoveries, Hospitalizations]
Index: []
```

[12]:

	Location	Cases	Deaths	Recoveries	Hospitalizations
1	Alabama	1659966	21138	509800	50767
2	Alaska	310531	1485	7165	4208
3	American Samoa	8326	34	3	0
4	Arizona	2611788	34545	0	157969
5	Arkansas	1039712	13787	992651	48032

[13]:
```
# Step5:        Fix casing or inconsistent values
# Standardize casing in the "Location" column
df["Location"] = df["Location"].str.strip().str.title()

# Replace common inconsistent values (e.g., NaN or "n/a")
df = df.replace(["NaN", "n/a", "N/A"], None)

print("Cleaned Data:")
print(df.head())
```

Cleaned Data:

	Location	Cases	Deaths	Recoveries	Hospitalizations
1	Alabama	1659966	21138	509800	50767
2	Alaska	310531	1485	7165	4208
3	American Samoa	8326	34	3	0
4	Arizona	2611788	34545	0	157969
5	Arkansas	1039712	13787	992651	48032

[14]:
```
# Step 6:        Conduct Fuzzy Matching
import warnings
# Suppress all warnings
warnings.filterwarnings("ignore")
from fuzzywuzzy import process

# Define a list of valid location names
valid_locations = df["Location"].unique()

# Example: Correct a list of misspelled names
location_to_fix = ["Califronia", "Florid", "Texs"]
for loc in location_to_fix:
    match = process.extractOne(loc, valid_locations)
    print(f"Best match for '{loc}' is '{match[0]}' with a confidence of␣
 ↪{match[1]}%")
```

```
Best match for 'Califronia' is 'California' with a confidence of 90%
Best match for 'Florid' is 'Florida[Vi]' with a confidence of 90%
Best match for 'Texs' is 'Texas' with a confidence of 89%
```

[15]:
```
# Final cleaned dataset
print("Cleaned Dataset:")
df.head()
```

```
Cleaned Dataset:
```

[15]:

	Location	Cases	Deaths	Recoveries	Hospitalizations
1	Alabama	1659966	21138	509800	50767
2	Alaska	310531	1485	7165	4208
3	American Samoa	8326	34	3	0
4	Arizona	2611788	34545	0	157969
5	Arkansas	1039712	13787	992651	48032

[16]:
```
# •        1 paragraph of the ethical implications of data wrangling specific␣
 ↪to your datasource and the steps you completed answering the following␣
 ↪questions:
# o        What changes were made to the data?
# o        Are there any legal or regulatory guidelines for your data or␣
 ↪project topic?
# o        What risks could be created based on the transformations done?
```

```
# o        Did you make any assumptions in cleaning/transforming the data?
# o        How was your data sourced / verified for credibility?
# o        Was your data acquired in an ethical way?
# o        How would you mitigate any of the ethical implications you have␣
 ↪identified?
```

Data wrangling for this dataset involved renaming columns for clarity, standardizing inconsistent casing, handling missing or invalid values, detecting and addressing outliers, removing duplicates, and conducting fuzzy matching to correct location names.

These changes, while necessary for readability and analysis, could inadvertently introduce bias or misrepresentations if assumptions (e.g., replacing missing values with zeros) are incorrect.

Depending on the dataset's context, such as public health or COVID-19 statistics, legal and regulatory guidelines like HIPAA or GDPR may govern its use, requiring compliance with privacy and ethical data handling standards.

Risks include potential misinterpretation of results if outliers are improperly handled or data is altered without adequate documentation. Assumptions were made about invalid entries (e.g., replacing "—" with 0), and these may not always reflect the original intent of the data.

The credibility of the data depends on its source, which must be verified for authenticity and ethical acquisition.

To mitigate risks, transparency in documenting cleaning steps, consulting domain experts to validate transformations, and ensuring the data aligns with regulatory guidelines are essential.

Additionally, clearly communicating the limitations and assumptions of the dataset prevents misuse or overinterpretation.