Assignment – Siddhartha Sen

An Introduction to Data Cleaning: Why It Matters

○ Discuss the importance of clean data for analysis and machine learning.

○ Provide examples of common issues like missing values and techniques for cleaning data.

<u>Answer :</u>

<u>Introduction :</u>

Data cleaning or also known as data wrangling refers to a variety of processes designed to transform raw data into more readily used formats to improve its quality and to make it more convenient to work with.

Data cleaning forms a major part of the Data Analysis pipeline. The pipeline is often represented as :

Acquistion → Cleaning → Representation → Analysis → Interpretation

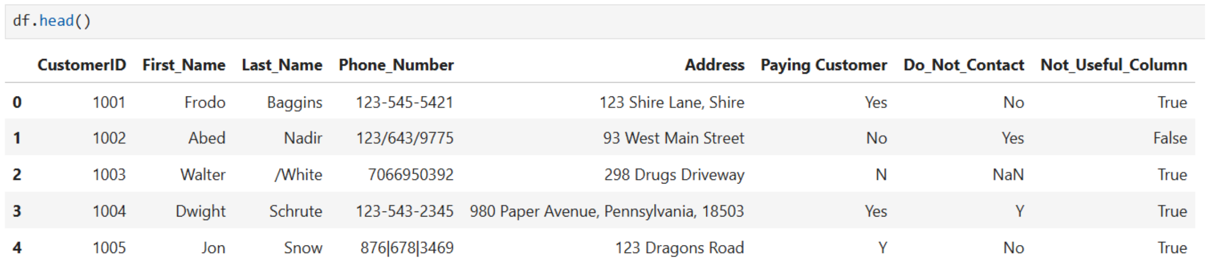
<u>Main Body :</u>

Various aspects prior and post to data cleaning involve :

1. **Discovery : It is when one collects data from various heterogenous sources and familiarizes oneself with the data collected.**
2. **Structuring : To transform the raw data into a form appropriate for the analysis we want to make.**
3. **Treating anomalies(Cleaning) : Tasks like standardising inputs, dealing with missing values, identifying and removing outliers are done in this part.**
4. **Enriching : To enrich the data (if needed) by adding data from other sources for eg : can be structured data like RDBMS or OORDBMS data or unstructured data sources like text files or semi structured data like XML files.**

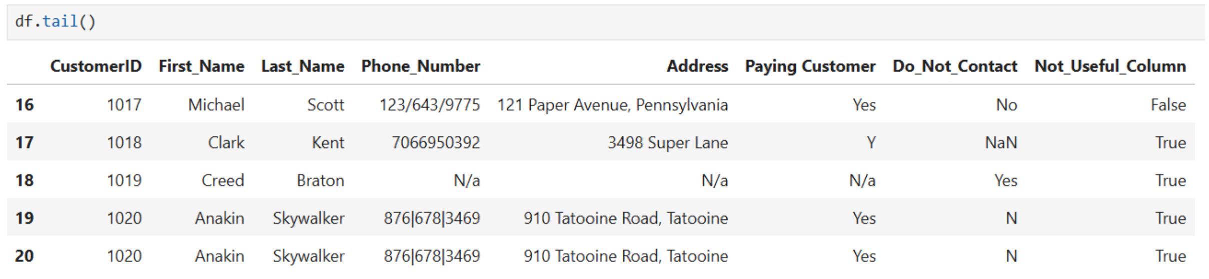
5. **Validating : To verify that our data is consistent (by checking data integrity, entity integrity, referential integrity etc) and of sufficient quality.**

Let us see an example from where we can have a basic understanding of dealing with data cleaning in real.



```
df.head()
```

	CustomerID	First_Name	Last_Name	Phone_Number	Address	Paying Customer	Do_Not_Contact	Not_Useful_Column
0	1001	Frodo	Baggins	123-545-5421	123 Shire Lane, Shire	Yes	No	True
1	1002	Abed	Nadir	123/643/9775	93 West Main Street	No	Yes	False
2	1003	Walter	/White	7066950392	298 Drugs Driveway	N	NaN	True
3	1004	Dwight	Schrute	123-543-2345	980 Paper Avenue, Pennsylvania, 18503	Yes	Y	True
4	1005	Jon	Snow	876|678|3469	123 Dragons Road	Y	No	True



```
df.tail()
```

	CustomerID	First_Name	Last_Name	Phone_Number	Address	Paying Customer	Do_Not_Contact	Not_Useful_Column
16	1017	Michael	Scott	123/643/9775	121 Paper Avenue, Pennsylvania	Yes	No	False
17	1018	Clark	Kent	7066950392	3498 Super Lane	Y	NaN	True
18	1019	Creed	Braton	N/a	N/a	N/a	Yes	True
19	1020	Anakin	Skywalker	876|678|3469	910 Tatooine Road, Tatooine	Yes	N	True
20	1020	Anakin	Skywalker	876|678|3469	910 Tatooine Road, Tatooine	Yes	N	True

The above represents some data from a call list. We can see that the Last_Name column has a '/' in one of its names. The Phone_Number column has different formats of representation some with a '-' in between while some with '/' or some with '|' etc. The Address column has maximum 3 address dimensions , having Street Address , State and Zip Code, all clubbed together. Also we can see the Paying Customer and Do_Not_Contact columns have null values as well as Yes and No represented sometimes as Y and N.

We need to *standardize* this data set so that the names are visible as just names (without any extra characters except letters). The Phone number must be in one particular format only (for eg we can have → ' …-…-…. ' as a uniform format for all the records or units). We can split the address into 3 sections as mentioned above. Also we need to represent the nominal columns of Paying Customer and Do_Not_Contact as Y and N OR Yes and No but not a mix of both. And as for the null values we can drop them off.

We can drop duplicates using the command :

df = df.drop_duplicates()

when we deal with a data frame in Pandas.

We can see from our data set that there exists a not useful column. We can drop it off.

```
df = df.drop(columns = 'Not_Useful_Column')
```

We can strip off the characters / , . , _ from the last name column using

```
df['Last_Name'] = df['Last_Name'].str.strip("./_")
```

We can replace all the characters(except a-z,A-Z and 0-9) that appear in the phone number with null.

```
df["Phone_Number"] = df["Phone_Number"].str.replace('[^a-zA-Z0-9]','',regex=True)
```

We can then format the phone number for having the ...-...-.... format. For that we need to change the entire column to string first

```
df["Phone_Number"]=df["Phone_Number"].apply(lambda x: str(x))
```

We use lambda function for the above operation.

```
df["Phone_Number"]=df["Phone_Number"].apply(lambda x: x[0:3]+'-'+x[3:6]+'-'+x[6:10])
```

We concatenate '-' after the first 3 characters and after the next 3 characters.

We replace the Na and nan in Phone number column with :

```
df["Phone_Number"] = df["Phone_Number"].str.replace('Na--','')
```

```
df["Phone_Number"] = df["Phone_Number"].str.replace('nan--','')
```

Splitting the address column into 3 sections :

```
df[["Street_Address","State","Zip_Code"]] = df["Address"].str.split(',',n = 2,expand=True)
```

n=2 means that the string in the "Address" column will be split at most twice on the delimiter , (a comma in this case).

Then we make the Yes and No represented uniformly as Y and N in the Paying Customer and Do_Not_Contact columns.

```
df["Paying Customer"] = df["Paying Customer"].str.replace("Yes","Y")
```

df["Paying Customer"] = df["Paying Customer"].str.replace("No","N")df["Do_Not_Contact"] =

df["Do_Not_Contact"].str.replace("Yes","Y")

df["Do_Not_Contact"] = df["Do_Not_Contact"].str.replace("No","N")

Some N/a values across the data frame can be replaced by :

```
df = df.replace('N/a','')
```

We can empty the NaN by :

```
df = df.fillna('')
```

If we need to filter out the list of Phone numbers we can call we can remove those where Do Not Contact is Y.

```
for x in df.index :
   if df.loc[x,"Do_Not_Contact"] == 'Y':
      df.drop(x,inplace=True)
```

We can remove the records having blank phone numbers :

```
for x in df.index :
   if df.loc[x,"Phone_Number"] == '':
      df.drop(x,inplace=True)
```

Removing the records for Do_Not_Contact column having blank data :

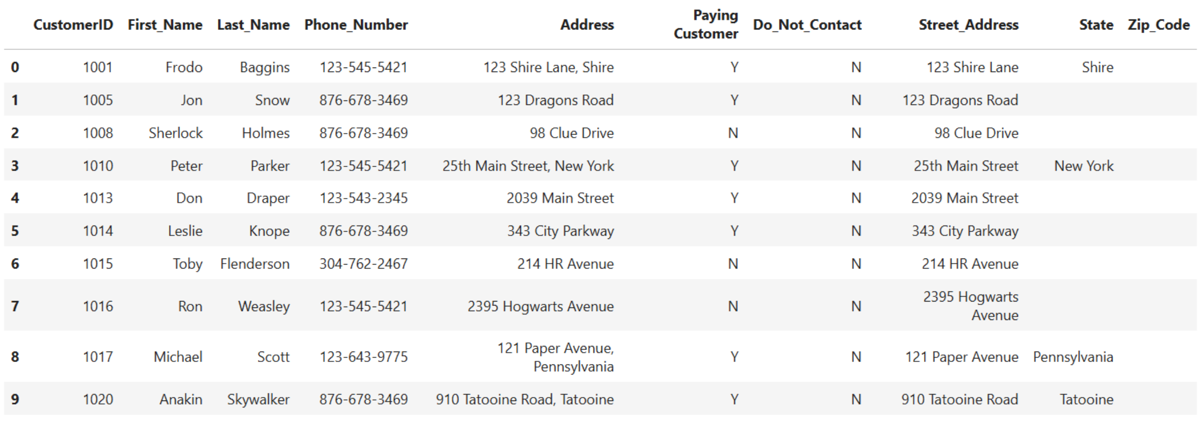
```
for x in df.index :
   if df.loc[x,"Do_Not_Contact"] == '':
      df.drop(x,inplace=True)
```

Finally we can reset the index and display the cleaned dataset that we have obtained:

```
df = df.reset_index(drop=True)

df
```

The Cleaned Dataset looks like :



	CustomerID	First_Name	Last_Name	Phone_Number	Address	Paying Customer	Do_Not_Contact	Street_Address	State	Zip_Code
0	1001	Frodo	Baggins	123-545-5421	123 Shire Lane, Shire	Y	N	123 Shire Lane	Shire	
1	1005	Jon	Snow	876-678-3469	123 Dragons Road	Y	N	123 Dragons Road		
2	1008	Sherlock	Holmes	876-678-3469	98 Clue Drive	N	N	98 Clue Drive		
3	1010	Peter	Parker	123-545-5421	25th Main Street, New York	Y	N	25th Main Street	New York	
4	1013	Don	Draper	123-543-2345	2039 Main Street	Y	N	2039 Main Street		
5	1014	Leslie	Knope	876-678-3469	343 City Parkway	Y	N	343 City Parkway		
6	1015	Toby	Flenderson	304-762-2467	214 HR Avenue	N	N	214 HR Avenue		
7	1016	Ron	Weasley	123-545-5421	2395 Hogwarts Avenue	N	N	2395 Hogwarts Avenue		
8	1017	Michael	Scott	123-643-9775	121 Paper Avenue, Pennsylvania	Y	N	121 Paper Avenue	Pennsylvania	
9	1020	Anakin	Skywalker	876-678-3469	910 Tatooine Road, Tatooine	Y	N	910 Tatooine Road	Tatooine	

<u>Conclusion:</u>

Proper data analysis is needed to generate trustworthy insights for appropriate decision making purposes in business enterprises. Data Cleaning plays a crucial role in determining the accuracy of the analysis. Without proper data cleaning, the insights obtained will be full of errors which can lead to wrong decision making. Insufficient and badly organized data can end up in underestimation or overestimation and may end up failing to identify correct patterns and trends.