Data Cleaning In Python

Common Issues With Data cleaning

- Reading the file
- Inconsistent Column Names
- Missing Data
- Different Data Types
- Duplicate rows
- and so on

In [95]:
```
import pandas as pd
import numpy as np
```

Loading or Reading the File

- Encoding Error
- Inconsistent rows

In [96]:
```
# problem 1 'utf-8
df = pd.read_csv("unclean_data.csv")
```

```
---------------------------------------------------------------------------
UnicodeDecodeError                        Traceback (most recent call last)
Input In [96], in <cell line: 2>()
      1 # problem 1 'utf-8
----> 2 df = pd.read_csv("unclean_data.csv")

File ~\anaconda3\lib\site-packages\pandas\util\_decorators.py:311, in depreca
te_nonkeyword_arguments.<locals>.decorate.<locals>.wrapper(*args, **kwargs)
    305 if len(args) > num_allow_args:
    306     warnings.warn(
    307         msg.format(arguments=arguments),
    308         FutureWarning,
    309         stacklevel=stacklevel,
    310     )
--> 311 return func(*args, **kwargs)

File ~\anaconda3\lib\site-packages\pandas\io\parsers\readers.py:680, in read_
csv(filepath_or_buffer, sep, delimiter, header, names, index_col, usecols, sq
ueeze, prefix, mangle_dupe_cols, dtype, engine, converters, true_values, fals
e_values, skipinitialspace, skiprows, skipfooter, nrows, na_values, keep_defa
ult_na, na_filter, verbose, skip_blank_lines, parse_dates, infer_datetime_for
mat, keep_date_col, date_parser, dayfirst, cache_dates, iterator, chunksize,
compression, thousands, decimal, lineterminator, quotechar, quoting, doublequ
ote, escapechar, comment, encoding, encoding_errors, dialect, error_bad_line
s, warn_bad_lines, on_bad_lines, delim_whitespace, low_memory, memory_map, fl
oat_precision, storage_options)
    665 kwds_defaults = _refine_defaults_read(
    666     dialect,
    667     delimiter,
   (...)
    676     defaults={"delimiter": ","},
    677 )
    678 kwds.update(kwds_defaults)
--> 680 return _read(filepath_or_buffer, kwds)

File ~\anaconda3\lib\site-packages\pandas\io\parsers\readers.py:575, in _read
(filepath_or_buffer, kwds)
    572 _validate_names(kwds.get("names", None))
    574 # Create the parser.
--> 575 parser = TextFileReader(filepath_or_buffer, **kwds)
    577 if chunksize or iterator:
    578     return parser

File ~\anaconda3\lib\site-packages\pandas\io\parsers\readers.py:933, in TextF
ileReader.__init__(self, f, engine, **kwds)
    930     self.options["has_index_names"] = kwds["has_index_names"]
    932 self.handles: IOHandles | None = None
--> 933 self._engine = self._make_engine(f, self.engine)

File ~\anaconda3\lib\site-packages\pandas\io\parsers\readers.py:1235, in Text
FileReader._make_engine(self, f, engine)
   1232     raise ValueError(msg)
   1234 try:
-> 1235     return mapping[engine](f, **self.options)
   1236 except Exception:
   1237     if self.handles is not None:
```

```
File ~\anaconda3\lib\site-packages\pandas\io\parsers\c_parser_wrapper.py:75,
in CParserWrapper.__init__(self, src, **kwds)
     72     kwds.pop(key, None)
     74 kwds["dtype"] = ensure_dtype_objs(kwds.get("dtype", None))
---> 75 self._reader = parsers.TextReader(src, **kwds)
     77 self.unnamed_cols = self._reader.unnamed_cols
     79 # error: Cannot determine type of 'names'

File ~\anaconda3\lib\site-packages\pandas\_libs\parsers.pyx:544, in pandas._l
ibs.parsers.TextReader.__cinit__()

File ~\anaconda3\lib\site-packages\pandas\_libs\parsers.pyx:633, in pandas._l
ibs.parsers.TextReader._get_header()

File ~\anaconda3\lib\site-packages\pandas\_libs\parsers.pyx:847, in pandas._l
ibs.parsers.TextReader._tokenize_rows()

File ~\anaconda3\lib\site-packages\pandas\_libs\parsers.pyx:1952, in pandas._
libs.parsers.raise_parser_error()

UnicodeDecodeError: 'utf-8' codec can't decode byte 0xff in position 275: inv
alid start byte
```

In [97]:
```
# Solution 1
# UTF Encoding
df1 = pd.read_csv("unclean_data.csv",encoding='latin1')
# we can set utf-8 in dataset as well to fix it
```

In [98]:
```
df1.head()
```

Out[98]:

	movie_title	num_critic_for_reviews	duration	DIRECTOR_facebook_likes	actor_3_facebook_like
0	Avatar?ÿ	723	178.0	10	85
1	Pirates of the Caribbean: At World's End?ÿ	302	NaN	563	100
2	Spectre?ÿ	602	148.0	20	16
3	The Dark Knight Rises?ÿ	813	NaN	22000	230
4	John Carter?ÿ	462	132.0	"475"	53

In [99]:
```
# Solution 2
# Use Text Editor and Save it as Utf-8,ISO-8859-1,latin1
df = pd.read_csv("unclean_data1.csv",encoding='utf8')
```

In [100]:
```
df.head()
```

Out[100]:

	movie_title	num_critic_for_reviews	duration	DIRECTOR_facebook_likes	actor_3_facebook_like
0	Avatar?ÿ	723	178.0	10	85
1	Pirates of the Caribbean: At World's End?ÿ	302	NaN	563	10
2	Spectre?ÿ	602	148.0	20	16
3	The Dark Knight Rises?ÿ	813	NaN	22000	230
4	John Carter?ÿ	462	132.0	"475"	53

Inconsistent Column Names

- Change Cases
- Rename them

Change the case to Upper

In [101]:
```
df.columns
```

Out[101]:
```
Index(['movie_title', 'num_critic_for_reviews', 'duration',
       'DIRECTOR_facebook_likes', 'actor_3_facebook_likes',
       'ACTOR_1_facebook_likes', 'gross', 'num_voted_users',
       'Cast_Total_facebook_likes', 'facenumber_in_poster',
       'num_user_for_reviews', 'budget', 'title_year',
       'ACTOR_2_facebook_likes', 'imdb_score', 'title_year.1'],
      dtype='object')
```

In [ ]:

In [ ]:

In [102]:
```
df.columns = df.columns.str.upper()
```

In [103]:
```
df.columns
```

Out[103]:
```
Index(['MOVIE_TITLE', 'NUM_CRITIC_FOR_REVIEWS', 'DURATION',
       'DIRECTOR_FACEBOOK_LIKES', 'ACTOR_3_FACEBOOK_LIKES',
       'ACTOR_1_FACEBOOK_LIKES', 'GROSS', 'NUM_VOTED_USERS',
       'CAST_TOTAL_FACEBOOK_LIKES', 'FACENUMBER_IN_POSTER',
       'NUM_USER_FOR_REVIEWS', 'BUDGET', 'TITLE_YEAR',
       'ACTOR_2_FACEBOOK_LIKES', 'IMDB_SCORE', 'TITLE_YEAR.1'],
      dtype='object')
```

Renaming Columns

In [104]:
```
df.rename(columns = {'DURATION':'TIME'})
df.columns
```

Out[104]:
```
Index(['MOVIE_TITLE', 'NUM_CRITIC_FOR_REVIEWS', 'DURATION',
       'DIRECTOR_FACEBOOK_LIKES', 'ACTOR_3_FACEBOOK_LIKES',
       'ACTOR_1_FACEBOOK_LIKES', 'GROSS', 'NUM_VOTED_USERS',
       'CAST_TOTAL_FACEBOOK_LIKES', 'FACENUMBER_IN_POSTER',
       'NUM_USER_FOR_REVIEWS', 'BUDGET', 'TITLE_YEAR',
       'ACTOR_2_FACEBOOK_LIKES', 'IMDB_SCORE', 'TITLE_YEAR.1'],
      dtype='object')
```

Missing Data

- Add a default value for missing data or use mean to fill it
- Delete the row/column with missing data
- Interpolate the rows
- Replace

To check for missing data

False means no missing data

- df.isnull().sum() int
- df.isnull().any() bool

In [105]:
```
df.isnull()
```

Out[105]:

	MOVIE_TITLE	NUM_CRITIC_FOR_REVIEWS	DURATION	DIRECTOR_FACEBOOK_LIKES	ACTC
0	False	False	False	False	
1	False	False	True	False	
2	False	False	False	False	
3	False	False	True	False	
4	False	False	False	False	
5	False	False	False	False	
6	False	False	True	False	
7	False	False	False	False	
8	False	False	False	False	
9	False	False	False	False	
10	False	False	False	True	
11	False	False	False	True	
12	False	False	False	False	
13	False	False	False	False	

In [106]:
```
df.isnull().any()
```

Out[106]:
```
MOVIE_TITLE                 False
NUM_CRITIC_FOR_REVIEWS      False
DURATION                     True
DIRECTOR_FACEBOOK_LIKES      True
ACTOR_3_FACEBOOK_LIKES      False
ACTOR_1_FACEBOOK_LIKES      False
GROSS                       False
NUM_VOTED_USERS              True
CAST_TOTAL_FACEBOOK_LIKES    True
FACENUMBER_IN_POSTER         True
NUM_USER_FOR_REVIEWS        False
BUDGET                      False
TITLE_YEAR                  False
ACTOR_2_FACEBOOK_LIKES       True
IMDB_SCORE                  False
TITLE_YEAR.1                 True
dtype: bool
```

In [ ]:

In [107]:
```
# For entire DataFrame
df.isnull().any().any()
```

Out[107]:
```
True
```

In [108]:
```
# Columns with NAN using Integer
df.isnull().sum()
```

Out[108]:
```
MOVIE_TITLE                  0
NUM_CRITIC_FOR_REVIEWS       0
DURATION                     3
DIRECTOR_FACEBOOK_LIKES      2
ACTOR_3_FACEBOOK_LIKES       0
ACTOR_1_FACEBOOK_LIKES       0
GROSS                        0
NUM_VOTED_USERS              1
CAST_TOTAL_FACEBOOK_LIKES    2
FACENUMBER_IN_POSTER         5
NUM_USER_FOR_REVIEWS         0
BUDGET                       0
TITLE_YEAR                   0
ACTOR_2_FACEBOOK_LIKES       1
IMDB_SCORE                   0
TITLE_YEAR.1                 7
dtype: int64
```

In [109]:
```
# Total Number of Missing NA
df.isnull().sum().sum()
```

Out[109]:
```
21
```

Adding A Default Value or Filling the Missing Data

In [110]:
```
df.head(5)
```

Out[110]:

	MOVIE_TITLE	NUM_CRITIC_FOR_REVIEWS	DURATION	DIRECTOR_FACEBOOK_LIKES	ACTOF
0	Avatar?ÿ	723	178.0	10	
1	Pirates of the Caribbean: At World's End?ÿ	302	NaN	563	
2	Spectre?ÿ	602	148.0	20	
3	The Dark Knight Rises?ÿ	813	NaN	22000	
4	John Carter?ÿ	462	132.0	"475"	

In [111]:
```
df_with_0 = df.fillna(0)
```

Different example

import pandas as pd import numpy as np

df = pd.DataFrame({'A': [1, np.nan, np.nan], 'B': [np.nan, 2, np.nan], 'C': [3, 4, np.nan]})

df_with_0 = df.fillna(0)

Print the original and new DataFrames print(df) print(df_with_0)

In [112]:
```
df_with_0.head()
```

Out[112]:

	MOVIE_TITLE	NUM_CRITIC_FOR_REVIEWS	DURATION	DIRECTOR_FACEBOOK_LIKES	ACTOI
0	Avatar?ÿ	723	178.0	10	
1	Pirates of the Caribbean: At World's End?ÿ	302	0.0	563	
2	Spectre?ÿ	602	148.0	20	
3	The Dark Knight Rises?ÿ	813	0.0	22000	
4	John Carter?ÿ	462	132.0	"475"	

Fill it with the mean

In [113]:
```
# Fill it with the mean
df['DURATION'].mean()
```

Out[113]:
```
150.72727272727272
```

In [114]:
```
df_with_mean = df.DURATION.fillna(df['DURATION'].mean())
```

In [115]:
```
df_with_mean
```

Out[115]:
```
0     178.000000
1     150.727273
2     148.000000
3     150.727273
4     132.000000
5     156.000000
6     150.727273
7     141.000000
8     141.000000
9     153.000000
10    183.000000
11    169.000000
12    106.000000
13    151.000000
Name: DURATION, dtype: float64
```

Droping NA

In [116]:
```
## Droping NA
df.head()
```

Out[116]:

	MOVIE_TITLE	NUM_CRITIC_FOR_REVIEWS	DURATION	DIRECTOR_FACEBOOK_LIKES	ACTOI
0	Avatar?ÿ	723	178.0	10	
1	Pirates of the Caribbean: At World's End?ÿ	302	NaN	563	
2	Spectre?ÿ	602	148.0	20	
3	The Dark Knight Rises?ÿ	813	NaN	22000	
4	John Carter?ÿ	462	132.0	"475"	

In [117]:
```
df.isnull().sum().sum()
```

Out[117]:
```
21
```

In [118]:
```
df.shape
```

Out[118]:
```
(14, 16)
```

In [119]:
```
df_drop = df.dropna()
```

In [120]:
```
df_drop.shape
```
Out[120]:
```
(4, 16)
```

In [121]:
```
df_drop.head()
```
Out[121]:

	MOVIE_TITLE	NUM_CRITIC_FOR_REVIEWS	DURATION	DIRECTOR_FACEBOOK_LIKES	ACTC
2	Spectre?ÿ	602	148.0	20	
8	Avengers: Age of Ultron?ÿ	635	141.0	10	
12	Quantum of Solace?ÿ	403	106.0	395	
13	Pirates of the Caribbean: Dead Man's Chest?ÿ	313	151.0	563	

Since thresh is set to 0, it means that any row that has at least one non-null value will be retained in the DataFrame. Essentially, this method does not drop any rows from the DataFrame based on null values.

In [122]:
```
df_drop_with_condition = df.dropna(thresh=0)
df_drop_with_condition
```

Out[122]:

	MOVIE_TITLE	NUM_CRITIC_FOR_REVIEWS	DURATION	DIRECTOR_FACEBOOK_LIKES	ACTC
0	Avatar?ÿ	723	178.0	10	
1	Pirates of the Caribbean: At World's End?ÿ	302	NaN	563	
2	Spectre?ÿ	602	148.0	20	
3	The Dark Knight Rises?ÿ	813	NaN	22000	
4	John Carter?ÿ	462	132.0	"475"	
5	Spider-Man 3?ÿ	392	156.0	23	
6	Tangled?ÿ	324	NaN	15	
7	Avengers: Age of Ultron?ÿ	635	141.0	10	
8	Avengers: Age of Ultron?ÿ	635	141.0	10	
9	Harry Potter and the Half-Blood Prince?ÿ	375	153.0	282	
10	Batman v Superman: Dawn of Justice?ÿ	673	183.0	NaN	
11	Superman Returns?ÿ	434	169.0	NaN	
12	Quantum of Solace?ÿ	403	106.0	395	
13	Pirates of the Caribbean: Dead Man's Chest?ÿ	313	151.0	563	

In [123]:
```
df_drop_with_condition.shape
```

Out[123]:
```
(14, 16)
```

In [124]:
```
df.shape
```

Out[124]:
```
(14, 16)
```

In [125]:
```
df_drop_column = df.dropna(axis=1)
#The dropna() method is used to remove missing or null values from a DataFrame
#In this case, we are removing any column that has at least one null value.
```

In [126]:
```
df_drop_column.shape
```

Out[126]:
```
(14, 9)
```

In [ ]:

In [ ]:

DATA CLEANING IN PYTHON

Dropping Duplicates

- drop_duplicates()
- keep='first'

In [127]:
```
df = pd.read_csv("unclean_data1.csv",encoding='utf8')
```

In [128]:
```
df.head(5)
```

Out[128]:

	movie_title	num_critic_for_reviews	duration	DIRECTOR_facebook_likes	actor_3_facebook_like
0	Avatar?ÿ	723	178.0	10	85
1	Pirates of the Caribbean: At World's End?ÿ	302	NaN	563	100
2	Spectre?ÿ	602	148.0	20	16
3	The Dark Knight Rises?ÿ	813	NaN	22000	2300
4	John Carter?ÿ	462	132.0	"475"	53

In [129]:
```
df.duplicated()
```

Out[129]:
```
0     False
1     False
2     False
3     False
4     False
5     False
6     False
7     False
8     False
9     False
10    False
11    False
12    False
13    False
dtype: bool
```

In [131]:
```
df.duplicated('movie_title')
```

Out[131]:
```
0     False
1     False
2     False
3     False
4     False
5     False
6     False
7     False
8      True
9     False
10    False
11    False
12    False
13    False
dtype: bool
```

In [132]:
```
df.head(10)
```

Out[132]:

	movie_title	num_critic_for_reviews	duration	DIRECTOR_facebook_likes	actor_3_facebook_like
0	Avatar?ÿ	723	178.0	10	85
1	Pirates of the Caribbean: At World's End?ÿ	302	NaN	563	100
2	Spectre?ÿ	602	148.0	20	16
3	The Dark Knight Rises?ÿ	813	NaN	22000	2300
4	John Carter?ÿ	462	132.0	"475"	53
5	Spider-Man 3?ÿ	392	156.0	23	400
6	Tangled?ÿ	324	NaN	15	28
7	Avengers: Age of Ultron?ÿ	635	141.0	10	1900
8	Avengers: Age of Ultron?ÿ	635	141.0	10	1900
9	Harry Potter and the Half-Blood Prince?ÿ	375	153.0	282	1000

In [133]:
```
df.shape
```

Out[133]:
```
(14, 16)
```

In [134]:
```
df_drop_dup = df.drop_duplicates('movie_title')
```

In [135]:
```
df_drop_dup.shape
```

Out[135]:
```
(13, 16)
```

Data Types Inconsistencies

- Change datatype after reading the csv
- Change datatype before reading the csv
-
  - pd.read_csv(url, dtype={'column1':float})

import pandas as pd

Read CSV file with column1 as float

df = pd.read_csv('data.csv', dtype={'column1': float})

Print the DataFrame

print(df)

```
In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:
```