Assignment 2 : Data Analysis with pandas and Visualization Libraries

Name: Navneet Yadav

Roll no. 2301560044

Program: MCA-I

Subject: AI-ML

Question 3

- GitHub Link: https://github.com/navYadav20/AI-ML- (https://github.com/navYadav20/AI-ML-)

Data cleaning

1. String Matching

In [75]:
```
from fuzzywuzzy import fuzz, process
```

In [8]:
```
pip install fuzzywuzzy
```

```
Collecting fuzzywuzzy
  Downloading fuzzywuzzy-0.18.0-py2.py3-none-any.whl (18 kB)
Installing collected packages: fuzzywuzzy
Successfully installed fuzzywuzzy-0.18.0
Note: you may need to restart the kernel to use updated packages.
```

In [1]:
```
from fuzzywuzzy import fuzz, process
```

```
C:\Users\msuse\anaconda3\Lib\site-packages\fuzzywuzzy\fuzz.py:11: UserWarn
ing: Using slow pure-python SequenceMatcher. Install python-Levenshtein to
remove this warning
  warnings.warn('Using slow pure-python SequenceMatcher. Install python-Le
venshtein to remove this warning')
```

In [2]:
```
berlin = ['Berlin, Germany',
          'Berlin, Deutschland',
          'Berlin',
          'Berlin, DE']
```

Try matching the first and second strings: 'Berlin, Germany' and 'Berlin, Deutschland'

In [3]:
```
fuzz.partial_ratio(berlin[0], berlin[1])
```

Out[3]: 60

In [4]:
```
fuzz.ratio?
```

In [5]:
```
fuzz.ratio(berlin[0], berlin[1])
```

Out[5]: 65

In [6]:
```
fuzz.token_set_ratio(berlin[0], berlin[1])
```

Out[6]: 62

Try matching the second and third strings: 'Berlin, Deutschland' and 'Berlin'

In [8]:
```
fuzz.partial_ratio(berlin[1], berlin[2])
```

Out[8]: 100

In [9]:
```
fuzz.ratio(berlin[1], berlin[2])
```

Out[9]: 48

In [10]:
```
fuzz.token_sort_ratio(berlin[1], berlin[2])
```

Out[10]: 50

What do you think will score lowest and highest for the final two elements:

- 'Berlin'
- 'Berlin, DE'

In [11]:
```
fuzz.token_set_ratio(berlin[2], berlin[3])
```

Out[11]: 100

Extracting a guess out of a list

In [12]:
```
choices = ['Germany', 'Deutschland', 'France',
           'United Kingdom', 'Great Britain',
           'United States']
```

In [13]:
```
process.extract('DE', choices, limit=2)
```

Out[13]:
```
[('Deutschland', 90), ('United States', 57)]
```

In [14]:
```
process.extract('UK', choices)
```

Out[14]:
```
[('Deutschland', 45),
 ('United Kingdom', 45),
 ('United States', 45),
 ('Germany', 0),
 ('France', 0)]
```

In [15]:
```
process.extract('frankreich', choices)
```

Out[15]:
```
[('France', 62),
 ('Great Britain', 41),
 ('Germany', 35),
 ('United Kingdom', 25),
 ('United States', 25)]
```

Will this properly extract?

In [16]:
```
process.extract('U.S.', choices)
```

Out[16]:
```
[('United States', 86),
 ('Deutschland', 60),
 ('United Kingdom', 57),
 ('Great Britain', 30),
 ('Germany', 0)]
```

In [ ]:
```

```

2. Managing Nulls with Pandas

In [18]:
```
import pandas as pd
from numpy import random
df = pd.read_csv('iot_example_with_nulls.csv')
```

Data Quality Check

In [19]:
```
df.head()
```

Out[19]:

	timestamp	username	temperature	heartrate	build	latest	note
0	2017-01-01T12:00:23	michaelsmith	12.0	67	4e6a7805-8faa-2768-6ef6-eb3198b483ac	0.0	interval
1	2017-01-01T12:01:09	kharrison	6.0	78	7256b7b0-e502-f576-62ec-ed73533c9c84	0.0	wake
2	2017-01-01T12:01:34	smithadam	5.0	89	9226c94b-bb4b-a6c8-8e02-cb42b53e9c90	0.0	NaN
3	2017-01-01T12:02:09	eddierodriguez	28.0	76	NaN	0.0	update
4	2017-01-01T12:02:36	kenneth94	29.0	62	122f1c6a-403c-2221-6ed1-b5caa08f11e0	NaN	NaN

In [22]:
```
df.dtypes
```

Out[22]:
```
timestamp       object
username        object
temperature    float64
heartrate        int64
build           object
latest         float64
note            object
dtype: object
```

In [23]:
```
df.note.value_counts()
```

Out[23]:
```
note
wake        16496
user        16416
interval    16274
sleep       16226
update      16213
test        16068
Name: count, dtype: int64
```

Let's remove all null values (including the note: n/a)

In [24]:
```
df = pd.read_csv('iot_example_with_nulls.csv',na_values=['n/a'])
```

Test to see if we can use dropna

In [25]:
```
df.shape
```

Out[25]:
```
(146397, 7)
```

In [26]:
```
df.dropna().shape
```

Out[26]:
```
(46116, 7)
```

In [27]:
```
df.dropna(how='all', axis=1).shape
```

Out[27]:
```
(146397, 7)
```

Test to see if we can drop columns

In [28]:
```
my_columns = list(df.columns)
my_columns
```

Out[28]:
```
['timestamp',
 'username',
 'temperature',
 'heartrate',
 'build',
 'latest',
 'note']
```

In [29]:
```
list(df.dropna(thresh=int(df.shape[0] * .9), axis=1).columns)
```

Out[29]:
```
['timestamp', 'username', 'heartrate']
```

I want to find all columns that have missing data

In [30]:
```
missing_info = list(df.columns[df.isnull().any()])
missing_info
```

Out[30]:
```
['temperature', 'build', 'latest', 'note']
```

In [31]:
```
for col in missing_info:
    num_missing = df[df[col].isnull() == True].shape[0]
    print('number missing for column {}: {}'.format(col,num_missing))
```

```
number missing for column temperature: 32357
number missing for column build: 32350
number missing for column latest: 32298
number missing for column note: 48704
```

In [32]:
```
for col in missing_info:
    percent_missing = df[df[col].isnull() == True].shape[0] / df.shape[0]
    print('percent missing for column {}: {}'.format(col, percent_missing))
```

```
percent missing for column temperature: 0.22102228870810195
percent missing for column build: 0.22097447352063226
percent missing for column latest: 0.22061927498514314
percent missing for column note: 0.332684412931959
```

Can I easily substitute majority values in for missing data?

In [33]:
```
df.note.value_counts()
```

Out[33]:
```
note
wake        16496
user        16416
interval    16274
sleep       16226
update      16213
test        16068
Name: count, dtype: int64
```

In [36]:
```
df.build.value_counts().head()
```

Out[36]:
```
build
4e6a7805-8faa-2768-6ef6-eb3198b483ac    1
12aefc6b-272c-751e-6117-134ee73e2649    1
fd4049c3-2297-14ac-a27e-6da57129dd10    1
0bcfab8f-bc25-3f8f-8585-0614e1555fd1    1
b0de05dd-2860-abbb-8be6-f5c0e30ca063    1
Name: count, dtype: int64
```

In [37]:
```
df.latest.value_counts()
```

Out[37]:
```
latest
0.0    75735
1.0    38364
Name: count, dtype: int64
```

In [39]:
```
df.latest = df.latest.fillna(0)
```

Have not yet addressed temperature missing values... Let's find a way to fill

In [40]:
```
df.username.value_counts().head()
df = df.set_index('timestamp')
df.head()
```

Out[40]:

timestamp	username	temperature	heartrate	build	latest	note
2017-01-01T12:00:23	michaelsmith	12.0	67	4e6a7805-8faa-2768-6ef6-eb3198b483ac	0.0	interval
2017-01-01T12:01:09	kharrison	6.0	78	7256b7b0-e502-f576-62ec-ed73533c9c84	0.0	wake
2017-01-01T12:01:34	smithadam	5.0	89	9226c94b-bb4b-a6c8-8e02-cb42b53e9c90	0.0	NaN
2017-01-01T12:02:09	eddierodriguez	28.0	76	NaN	0.0	update
2017-01-01T12:02:36	kenneth94	29.0	62	122f1c6a-403c-2221-6ed1-b5caa08f11e0	0.0	NaN

In [41]:
```
df.temperature = df.groupby('username').temperature.fillna(method='backfill
```

```
C:\Users\msuse\AppData\Local\Temp\ipykernel_11700\2390181958.py:1: FutureW
arning: Series.fillna with 'method' is deprecated and will raise in a futu
re version. Use obj.ffill() or obj.bfill() instead.
  df.temperature = df.groupby('username').temperature.fillna(
```

In [42]:
```
df.temperature
```

Out[42]:
```
timestamp
2017-01-01T12:00:23    12.0
2017-01-01T12:01:09     6.0
2017-01-01T12:01:34     5.0
2017-01-01T12:02:09    28.0
2017-01-01T12:02:36    29.0
                       ...
2017-02-28T23:58:06    15.0
2017-02-28T23:58:43     NaN
2017-02-28T23:59:23     NaN
2017-02-28T23:59:48    17.0
2017-03-01T00:00:30    23.0
Name: temperature, Length: 146397, dtype: float64
```

In [ ]:
```

```

3.Scikit Learn Preprocessing

In [50]:
```
from sklearn import preprocessing
from sklearn.impute import SimpleImputer
import pandas as pd
from datetime import datetime
```

In [51]:
```
hvac = pd.read_csv('HVAC_with_nulls.csv')
```

Checking data quality

In [45]:
```
hvac.dtypes
```

Out[45]:
```
Date           object
Time           object
TargetTemp    float64
ActualTemp      int64
System          int64
SystemAge     float64
BuildingID      int64
10            float64
dtype: object
```

In [46]:
```
hvac.shape
```

Out[46]:
```
(8000, 8)
```

In [47]:
```
hvac.head()
```

Out[47]:

	Date	Time	TargetTemp	ActualTemp	System	SystemAge	BuildingID	10
0	6/1/13	0:00:01	66.0	58	13	20.0	4	NaN
1	6/2/13	1:00:01	NaN	68	3	20.0	17	NaN
2	6/3/13	2:00:01	70.0	73	17	20.0	18	NaN
3	6/4/13	3:00:01	67.0	63	2	NaN	15	NaN
4	6/5/13	4:00:01	68.0	74	16	9.0	3	NaN

Impute missing values with mean

In [61]:
```
from sklearn.impute import SimpleImputer
```

In [62]:
```
imp = SimpleImputer(missing_values='NaN', strategy='mean')
```

In [63]:
```
hvac_numeric = hvac[['TargetTemp', 'SystemAge']]
```

In [67]:
```
hvac_numeric = hvac_numeric.fillna(hvac_numeric.mean())
```

Scale temperature values

In [71]:
```
hvac['ScaledTemp'] = preprocessing.scale(hvac['ActualTemp'])
hvac['ScaledTemp'].head()
```

Out[71]:
```
0   -1.293272
1    0.048732
2    0.719733
3   -0.622270
4    0.853934
Name: ScaledTemp, dtype: float64
```

Scale using a min and max scaler

In [74]:
```
min_max_scaler = preprocessing.MinMaxScaler()
```

In [73]:
```
temp_minmax = min_max_scaler.fit_transform(hvac[['ActualTemp']])
temp_minmax
```

Out[73]:
```
array([[0.12],
       [0.52],
       [0.72],
       ...,
       [0.56],
       [0.32],
       [0.44]])
```