Data Preprocessing

Following are Data Preprocessing Steps include in this Notebook:

- Standardization
- Encoding
- Disecretization
- Normalization

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
data = pd.read_csv("IOT-temp.csv")
```

```
data.head()
```

	id	room_id/id	noted_date	temp	out/in
0	__export__.temp_log_196134_bd201015	Room Admin	08-12-2018 09:30	29	In
1	__export__.temp_log_196131_7bca51bc	Room Admin	08-12-2018 09:30	29	In
2	__export__.temp_log_196127_522915e3	Room Admin	08-12-2018 09:29	41	Out
3	__export__.temp_log_196128_be0919cf	Room Admin	08-12-2018 09:29	41	Out
4	__export__.temp_log_196126_d30b72fb	Room Admin	08-12-2018 09:29	31	In

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 97606 entries, 0 to 97605
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   id          97606 non-null  object
 1   room_id/id  97606 non-null  object
 2   noted_date  97606 non-null  object
 3   temp        97606 non-null  int64 
 4   out/in      97606 non-null  object
dtypes: int64(1), object(4)
memory usage: 3.7+ MB
```

```
from sklearn.preprocessing import LabelEncoder , OneHotEncoder
```

```
data['out/in'].value_counts()
```

```
Out    77261
In     20345
```

```
Name: out/in, dtype: int64
```

```
le = LabelEncoder()
data['out/in'] = le.fit_transform(data['out/in'])
```

```
data['out/in'].value_counts()
```

```
1    77261
0    20345
Name: out/in, dtype: int64
```

```
le.classes_
```

```
array(['In', 'Out'], dtype=object)
```

```
data['temp'].value_counts()
```

```
39    10203
28     8831
29     7922
40     7798
31     7236
30     6614
37     5723
32     5408
27     4631
41     4354
36     3965
38     3867
42     3447
33     3437
34     2613
43     2004
44     1774
35     1582
45     1508
46     1201
47     1044
48      971
26      699
49      401
25      224
24       66
50       55
22       19
23        5
51        2
21        2
Name: temp, dtype: int64
```

```
one_hot = OneHotEncoder()
transformed_data = one_hot.fit_transform(data['temp'].values.reshape(-1,1)).toarray()
```

```
one_hot.categories_
```

```
[array([21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37,
        38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51])]
```

```
transformed_data = pd.DataFrame(transformed_data ,
                                columns = [21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32,
    38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51])
```

```
transformed_data.head()
```

3	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41
0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

```
transformed_data.iloc[50 , ]
```

```
21    0.0
22    0.0
23    0.0
24    0.0
25    0.0
26    0.0
27    0.0
28    0.0
29    0.0
30    1.0
31    0.0
32    0.0
33    0.0
34    0.0
35    0.0
36    0.0
37    0.0
38    0.0
39    0.0
40    0.0
41    0.0
42    0.0
43    0.0
44    0.0
45    0.0
46    0.0
47    0.0
48    0.0
49    0.0
50    0.0
```

```
51    0.0
Name: 50, dtype: float64
```

```
data['temp'][50]
```

```
30
```

Normalization & Standardization

```
numeric_columns = [c for c in data.columns if data[c].dtype != np.dtype('O')]
```

```
len(numeric_columns) , len(data.columns)
```

```
(3, 5)
```

```
temp_data = data[numeric_columns]
```

```
temp_data
```

	id	temp	out/in
0	46863	29	0
1	46862	29	0
2	46859	41	1
3	46860	41	1
4	46858	31	0
...	...	...	...
97601	92483	31	0
97602	27424	31	0
97603	239	31	0
97604	13183	31	0
97605	19367	31	0

97606 rows × 3 columns

```
from sklearn.preprocessing import StandardScaler , MinMaxScaler
```

Normalization

```
import warnings
warnings.filterwarnings('ignore')
```

```
normalizer = MinMaxScaler()
```

```
temp_data.dropna(axis = 1 , inplace = True)
```

```
normalized_data = normalizer.fit_transform(temp_data)
```

```
pd.DataFrame(normalized_data , columns = temp_data.columns)
```

	id	temp	out/in
0	0.480134	0.266667	0.0
1	0.480124	0.266667	0.0
2	0.480093	0.666667	1.0
3	0.480103	0.666667	1.0
4	0.480083	0.333333	0.0
...	...	...	...
97601	0.947533	0.333333	0.0
97602	0.280972	0.333333	0.0
97603	0.002449	0.333333	0.0
97604	0.135066	0.333333	0.0
97605	0.198424	0.333333	0.0

97606 rows × 3 columns

Standardization

```
standard_scaler = StandardScaler()
```

```
standardized_data = standard_scaler.fit_transform(temp_data)
```

```
pd.DataFrame(standardized_data , columns = temp_data.columns)
```

	id	temp	out/in
0	-0.068817	-1.062131	-1.948728
1	-0.068852	-1.062131	-1.948728
2	-0.068959	1.043207	0.513155
3	-0.068923	1.043207	0.513155
4	-0.068994	-0.711241	-1.948728
...	...	...	...
97601	1.550292	-0.711241	-1.948728
97602	-0.758730	-0.711241	-1.948728
97603	-1.723559	-0.711241	-1.948728
97604	-1.264161	-0.711241	-1.948728
97605	-1.044683	-0.711241	-1.948728

97606 rows × 3 columns

Handling With Missing Values

```
data.isnull().sum()
```

```
id            0
room_id/id    0
noted_date    0
temp          0
out/in        0
dtype: int64
```

Discretization

```
from sklearn.preprocessing import KBinsDiscretizer
```

```
temp_data.head()
```

Quantile Discretization Transform

```
1  46862    29       0
```

```
trans = KBinsDiscretizer(n_bins =10 , encode = 'ordinal' , strategy='quantile')
new_data = trans.fit_transform(temp_data)
```

```
pd.DataFrame(new_data,columns = temp_data.columns )
```

	id	temp	out/in
0	4.0	2.0	0.0
1	4.0	2.0	0.0
2	4.0	8.0	0.0
3	4.0	8.0	0.0
4	4.0	3.0	0.0
...	...	...	...
97601	9.0	3.0	0.0
97602	2.0	3.0	0.0
97603	0.0	3.0	0.0
97604	1.0	3.0	0.0
97605	1.0	3.0	0.0

97606 rows × 3 columns

	id	temp	out/in
0	4.0	2.0	0.0
1	4.0	2.0	0.0
2	4.0	8.0	0.0
3	4.0	8.0	0.0
4	4.0	3.0	0.0
...	...	...	...
97601	9.0	3.0	0.0
97602	2.0	3.0	0.0
97603	0.0	3.0	0.0
97604	1.0	3.0	0.0
97605	1.0	3.0	0.0

97606 rows × 3 columns

Uniform Discretization Transform

```
trans = KBinsDiscretizer(n_bins =10 , encode = 'ordinal' , strategy='uniform')
new_data = trans.fit_transform(temp_data)

pd.DataFrame(new_data,columns = temp_data.columns )
```

	id	temp	out/in
0	4.0	2.0	0.0
1	4.0	2.0	0.0
2	4.0	6.0	9.0
3	4.0	6.0	9.0
4	4.0	3.0	0.0
...	...	...	...
97601	9.0	3.0	0.0
97602	2.0	3.0	0.0
97603	0.0	3.0	0.0
97604	1.0	3.0	0.0
97605	1.0	3.0	0.0

97606 rows × 3 columns

KMeans Discretization Transform

```
trans = KBinsDiscretizer(n_bins =10 , encode = 'ordinal' , strategy='kmeans')
new_data = trans.fit_transform(temp_data)

pd.DataFrame(new_data,columns = temp_data.columns )
```

	id	temp	out/in
0	4.0	2.0	0.0
1	4.0	2.0	0.0
2	4.0	6.0	1.0
4	4.0	3.0	0.0
...	...	...	...
97601	9.0	3.0	0.0
97602	2.0	3.0	0.0
97603	0.0	3.0	0.0
97604	1.0	3.0	0.0
97605	1.0	3.0	0.0

97606 rows × 3 columns

✓ 0s completed at 2:19 PM