# Lab 6 – AEP_Normalization_OneHot_Cyclic

## Objective

The aim of this lab is to preprocess features in the AEP dataset by applying:

- **Normalization** – To scale numerical features
- **One-Hot Encoding** – To convert categorical variables
- **Cyclical Encoding** – To transform periodic features like hours or months for better machine learning model understanding

---

## Introduction

Data preprocessing plays a vital role in building robust and effective machine learning models. This lab focuses on preparing the AEP dataset by applying three common techniques:

1. **Normalization** scales features to a similar range (usually [0, 1] or [-1, 1]), preventing features with larger magnitudes from dominating model training.

2. **One-Hot Encoding** converts categorical features into a numerical format without assuming any ordinal relationship.

3. **Cyclical Encoding** is especially useful for periodic features (e.g., hours, months), converting them into sine and cosine values to preserve their circular nature.

---

## Tools Used

- **Language:** Python

- **Libraries:** `pandas`, `numpy`, `sklearn`, `matplotlib`

```
import sys
sys.path.append(r'C:\Users\PMLS\ML\LAB6\timeseires')

import numpy as np
import pandas as pd
from pandas import read_csv
import pickle
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn.preprocessing import OneHotEncoder
from timeseires.utils import t_v_t_split as sp
```

```python
df=pd.read_csv(r'C:\Users\PMLS\ML\LAB6\5_features_extracted.csv',
index_col=['Datetime'], parse_dates=['Datetime'])
df.head()
```

```
                         aep  year_day  holiday  weekend  winter
spring  \
Datetime

2004-10-01 01:00:00  12379.0       275        0        0       0
0
2004-10-01 02:00:00  11935.0       275        0        0       0
0
2004-10-01 03:00:00  11692.0       275        0        0       0
0
2004-10-01 04:00:00  11597.0       275        0        0       0
0
2004-10-01 05:00:00  11681.0       275        0        0       0
0

                     summer  fall  hour  month  day_of_week
Datetime
2004-10-01 01:00:00       0     1     1     10            4
2004-10-01 02:00:00       0     1     2     10            4
2004-10-01 03:00:00       0     1     3     10            4
2004-10-01 04:00:00       0     1     4     10            4
2004-10-01 05:00:00       0     1     5     10            4
```

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 121296 entries, 2004-10-01 01:00:00 to 2018-08-03
00:00:00
Data columns (total 11 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   aep          121296 non-null  float64
 1   year_day     121296 non-null  int64
 2   holiday      121296 non-null  int64
 3   weekend      121296 non-null  int64
 4   winter       121296 non-null  int64
 5   spring       121296 non-null  int64
 6   summer       121296 non-null  int64
 7   fall         121296 non-null  int64
 8   hour         121296 non-null  int64
 9   month        121296 non-null  int64
 10  day_of_week  121296 non-null  int64
dtypes: float64(1), int64(10)
memory usage: 11.1 MB
```

# Function to split data For Training, Validation & Test

# train, test & validation split

```
train_set , validation_set , test_set = sp.t_v_t(df,70,20,10)
```

```
print(train_set.shape)
print(validation_set.shape)
print(test_set.shape)
```

```
(84907, 11)
(24259, 11)
(12130, 11)
```

```
(84907+24259+12130)-121296
```

```
0
```

```
test_set
```

```
                         aep  year_day  holiday  weekend  winter  
spring  \
Datetime                                                          
                                                                  
2017-03-15 15:00:00  17979.0        74        0        0       0   
1
2017-03-15 16:00:00  17569.0        74        0        0       0   
1
2017-03-15 17:00:00  17445.0        74        0        0       0   
1
2017-03-15 18:00:00  17545.0        74        0        0       0   
1
2017-03-15 19:00:00  17713.0        74        0        0       0   
1
...                      ...       ...      ...      ...     ...     .
..
2018-08-02 20:00:00  17673.0       214        0        0       0   
0
2018-08-02 21:00:00  17303.0       214        0        0       0   
0
2018-08-02 22:00:00  17001.0       214        0        0       0   
0
2018-08-02 23:00:00  15964.0       214        0        0       0   
0
2018-08-03 00:00:00  14809.0       215        0        0       0   
0
```

```
                     summer  fall  hour  month  day_of_week
Datetime                                                    
2017-03-15 15:00:00       0     0    15      3            2
2017-03-15 16:00:00       0     0    16      3            2
2017-03-15 17:00:00       0     0    17      3            2
2017-03-15 18:00:00       0     0    18      3            2
2017-03-15 19:00:00       0     0    19      3            2
...                     ...   ...   ...    ...          ...
2018-08-02 20:00:00       1     0    20      8            3
2018-08-02 21:00:00       1     0    21      8            3
2018-08-02 22:00:00       1     0    22      8            3
2018-08-02 23:00:00       1     0    23      8            3
2018-08-03 00:00:00       1     0     0      8            4

[12130 rows x 11 columns]
```

# Train

MinMax

```
train_set_load              = train_set['aep'].values.reshape(-1, 1)

validation_set_load         = validation_set['aep'].values.reshape(-1, 
1)
test_set_load               = test_set['aep'].values.reshape(-1, 1)
#......................................................................
....
scaler = MinMaxScaler(feature_range=(0, 1))
scaler.fit(train_set_load)
#scaler = StandardScaler()
#scaler.fit(train_set_load)
#......................................................................
....
scaled_train_set_load      = scaler.transform(train_set_load)
scaled_validation_set_load = scaler.transform(validation_set_load)
scaled_test_set_load       = scaler.transform(test_set_load)
pickle.dump(scaler, open ("AEPscaler.pkl",'wb') )
scaled_train_set_load.shape
```

```
(84907, 1)
```

```
train_numerical = scaled_train_set_load
train_numerical.shape
```

```
(84907, 1)
```

```
train_set.describe()
```

```
                aep      year_day      holiday      weekend
winter  \
count  84907.000000  84907.000000  84907.000000  84907.000000
84907.000000
mean   15786.426325    182.050019      0.051162      0.286007
0.254961
std     2555.537364    106.809303      0.220329      0.451895
0.435842
min     9669.000000      1.000000      0.000000      0.000000
0.000000
25%    13948.000000     89.000000      0.000000      0.000000
0.000000
50%    15615.000000    179.000000      0.000000      0.000000
0.000000
75%    17464.000000    277.000000      0.000000      1.000000
1.000000
max    22556.000000    366.000000      1.000000      1.000000
1.000000

             spring        summer          fall          hour
month  \
count  84907.000000  84907.000000  84907.000000  84907.000000
84907.000000
mean       0.260049      0.236258      0.248731     11.499664
6.488676
std        0.438664      0.424785      0.432280      6.921974
3.492888
min        0.000000      0.000000      0.000000      0.000000
1.000000
25%        0.000000      0.000000      0.000000      5.500000
3.000000
50%        0.000000      0.000000      0.000000     11.000000
6.000000
75%        1.000000      0.000000      0.000000     17.000000
10.000000
max        1.000000      1.000000      1.000000     23.000000
12.000000

        day_of_week
count  84907.000000
mean       3.001543
std        2.000102
min        0.000000
25%        1.000000
50%        3.000000
75%        5.000000
max        6.000000
```

```
(12379.-9669.)/(22556.0-9669)
```

```
0.21028943896950414
```

```
train_numerical
```

```
array([[0.21028944],
       [0.17583611],
       [0.1569799 ],
       ...,
       [0.37991775],
       [0.38410802],
       [0.36633817]])
```

OneHot Encoding

```
df.columns
```

```
Index(['aep', 'year_day', 'holiday', 'weekend', 'winter', 'spring',
'summer',
       'fall', 'hour', 'month', 'day_of_week'],
      dtype='object')
```

```
_=df.columns
_[2],_[3]
```

```
('holiday', 'weekend')
```

```
train_set
```

```
                         aep  year_day  holiday  weekend  winter
spring  \
Datetime

2004-10-01 01:00:00  12379.0       275        0        0       0
0
2004-10-01 02:00:00  11935.0       275        0        0       0
0
2004-10-01 03:00:00  11692.0       275        0        0       0
0
2004-10-01 04:00:00  11597.0       275        0        0       0
0
2004-10-01 05:00:00  11681.0       275        0        0       0
0
...                      ...       ...      ...      ...     ...      .
..
2014-06-08 15:00:00  14420.0       159        0        1       0
0
2014-06-08 16:00:00  14498.0       159        0        1       0
0
2014-06-08 17:00:00  14565.0       159        0        1       0
0
2014-06-08 18:00:00  14619.0       159        0        1       0
```

```
0
2014-06-08 19:00:00  14390.0       159        0        1       0
0

                     summer  fall  hour  month  day_of_week
Datetime
2004-10-01 01:00:00       0     1     1     10            4
2004-10-01 02:00:00       0     1     2     10            4
2004-10-01 03:00:00       0     1     3     10            4
2004-10-01 04:00:00       0     1     4     10            4
2004-10-01 05:00:00       0     1     5     10            4
...                     ...   ...   ...    ...          ...
2014-06-08 15:00:00       1     0    15      6            6
2014-06-08 16:00:00       1     0    16      6            6
2014-06-08 17:00:00       1     0    17      6            6
2014-06-08 18:00:00       1     0    18      6            6
2014-06-08 19:00:00       1     0    19      6            6

[84907 rows x 11 columns]
```

```
train_setO     = train_set[:].values

holiday        = train_setO[:,2:3]
weekend        = train_setO[:,3:4]


en_holiday     = OneHotEncoder(handle_unknown='ignore')
en_weekend    = OneHotEncoder(handle_unknown='ignore')


holidayf       = en_holiday.fit(holiday)             #2
holidayt       = holidayf.transform(holiday).toarray()

weekendf       = en_weekend.fit(weekend)             #2
weekendt       = weekendf.transform(weekend).toarray()




train_categorical = np.concatenate((holidayt,weekendt), axis=1)
train_categorical.shape
```

```
(84907, 4)
```

Cyclic

```
df.columns
```

```
Index(['aep', 'year_day', 'holiday', 'weekend', 'winter', 'spring',
'summer',
       'fall', 'hour', 'month', 'day_of_week'],
      dtype='object')
```

```
cyclic_train = train_set[['month','day_of_week','hour','winter',
'spring', 'summer','fall','year_day']]
cyclic_train = cyclic_train[:].values


sin_montht   = np.sin(2*np.pi*cyclic_train[:,0:1]/12)
cos_montht   = np.cos(2*np.pi*cyclic_train[:,0:1]/12)

sin_weekt    = np.sin(2*np.pi*cyclic_train[:,1:2]/6)
cos_weekt    = np.cos(2*np.pi*cyclic_train[:,1:2]/6)

sin_hourt    = np.sin(2*np.pi*cyclic_train[:,2:3]/24)
cos_hourt    = np.cos(2*np.pi*cyclic_train[:,2:3]/24)

sin_wintert    = np.sin(2*np.pi*cyclic_train[:,3:4]/4)
cos_wintert    = np.cos(2*np.pi*cyclic_train[:,3:4]/4)

sin_springt    = np.sin(2*np.pi*cyclic_train[:,4:5]/4)
cos_springt    = np.cos(2*np.pi*cyclic_train[:,4:5]/4)

sin_summert    = np.sin(2*np.pi*cyclic_train[:,5:6]/4)
cos_summert    = np.cos(2*np.pi*cyclic_train[:,5:6]/4)

sin_fallt    = np.sin(2*np.pi*cyclic_train[:,6:7]/4)
cos_fallt    = np.cos(2*np.pi*cyclic_train[:,6:7]/4)

sin_year_dayt    = np.sin(2*np.pi*cyclic_train[:,7:8]/365)
cos_year_dayt    = np.cos(2*np.pi*cyclic_train[:,7:8]/365)

train_cyclic = np.concatenate((sin_montht, cos_montht,
                               sin_weekt, cos_weekt,
                               sin_hourt,cos_hourt,
                               sin_wintert,cos_wintert,
                               sin_springt,cos_springt,
                               sin_summert ,cos_summert,
                               sin_fallt,cos_fallt,
                              sin_year_dayt ,cos_year_dayt ), axis=1)

train = np.concatenate((train_numerical,train_categorical,
train_cyclic), axis=1)
train.shape
```

```
(84907, 21)
```

```
train_df = pd.DataFrame(data = train.transpose(), index = ['aep',

'Is_holiday1','Is_holiday2',
'Is_Weekend1','Is_Weekend2',

'sin_month', 'cos_month',

'sin_week','cos_week',
                                                                                         'sin_hour',
'cos_hour',

'sin_wintert','cos_wintert',

'sin_springt','cos_springt',

'sin_summert' ,'cos_summert',

'sin_fallt','cos_fallt',

'sin_year_dayt ','cos_year_dayt ']).transpose()
train_df.to_csv('7_AEP_train.csv', index=False)
```

# validation

MinMax

```
validation_numerical = scaled_validation_set_load
validation_numerical.shape
```

```
(24259, 1)
```

OneHot

```

validation_set0 = validation_set[:].values

holiday        = validation_set0[:,2:3]
weekend        = validation_set0[:,3:4]



holidayt       = holidayf.transform(holiday).toarray()

weekendt       = weekendf.transform(weekend).toarray()



validation_categorical = np.concatenate((holidayt,weekendt), axis=1)
validation_categorical.shape
```

```
(24259, 4)
```

cyclic

```
cyclic_validation =
validation_set[['month','day_of_week','hour','winter', 'spring',
'summer','fall','year_day']]
cyclic_validation = cyclic_validation[:].values

sin_montht   = np.sin(2*np.pi*cyclic_validation[:,0:1]/12)
cos_montht   = np.cos(2*np.pi*cyclic_validation[:,0:1]/12)

sin_weekt    = np.sin(2*np.pi*cyclic_validation[:,1:2]/6)
cos_weekt    = np.cos(2*np.pi*cyclic_validation[:,1:2]/6)

sin_hourt    = np.sin(2*np.pi*cyclic_validation[:,2:3]/24)
cos_hourt    = np.cos(2*np.pi*cyclic_validation[:,2:3]/24)

sin_wintert    = np.sin(2*np.pi*cyclic_validation[:,3:4]/4)
cos_wintert    = np.cos(2*np.pi*cyclic_validation[:,3:4]/4)

sin_springt    = np.sin(2*np.pi*cyclic_validation[:,4:5]/4)
cos_springt    = np.cos(2*np.pi*cyclic_validation[:,4:5]/4)

sin_summert    = np.sin(2*np.pi*cyclic_validation[:,5:6]/4)
cos_summert    = np.cos(2*np.pi*cyclic_validation[:,5:6]/4)

sin_fallt    = np.sin(2*np.pi*cyclic_validation[:,6:7]/4)
cos_fallt    = np.cos(2*np.pi*cyclic_validation[:,6:7]/4)

sin_year_dayt    = np.sin(2*np.pi*cyclic_validation[:,7:8]/365)
cos_year_dayt    = np.cos(2*np.pi*cyclic_validation[:,7:8]/365)

validation_cyclic = np.concatenate((sin_montht, cos_montht,
                               sin_weekt, cos_weekt,
                               sin_hourt,cos_hourt,
                               sin_wintert,cos_wintert,
                               sin_springt,cos_springt,
                               sin_summert ,cos_summert,

sin_fallt,cos_fallt,sin_year_dayt,cos_year_dayt), axis=1)

validation_cyclic.shape
```

```
(24259, 16)
```

```
validation =
np.concatenate((validation_numerical,validation_categorical,
validation_cyclic), axis=1)
validation.shape
```

```
(24259, 21)
```

```
validation_df = pd.DataFrame(data = validation.transpose(), index =
['aep',

'Is_holiday1','Is_holiday2',

'Is_Weekend1','Is_Weekend2',

'sin_month', 'cos_month',

'sin_week','cos_week',
                                                           'sin_hour',
'cos_hour',

'sin_wintert','cos_wintert',

'sin_springt','cos_springt',

'sin_summert' ,'cos_summert',

'sin_fallt','cos_fallt',

'sin_year_dayt','cos_year_dayt']).transpose()

validation_df.to_csv('8_AEP_validation.csv', index=False)

validation_df.shape
```

```
(24259, 21)
```

# Test

MinMax

```
test_numerical = scaled_test_set_load
test_numerical.shape
```

```
(12130, 1)
```

OneHot

```
test_set0 = test_set[:].values

holiday        = test_set0[:,2:3]
weekend        = test_set0[:,3:4]



holidayt       = holidayf.transform(holiday).toarray()
```

```
weekendt       = weekendf.transform(weekend).toarray()



test_categorical = np.concatenate((holidayt,weekendt), axis=1)
test_categorical.shape
```

```
(12130, 4)
```

cyclic

```
cyclic_test = test_set[['month','day_of_week','hour','winter',
'spring', 'summer','fall', 'year_day']]
cyclic_test = cyclic_test[:].values

sin_montht   = np.sin(2*np.pi*cyclic_test[:,0:1]/12)
cos_montht   = np.cos(2*np.pi*cyclic_test[:,0:1]/12)

sin_weekt    = np.sin(2*np.pi*cyclic_test[:,1:2]/6)
cos_weekt    = np.cos(2*np.pi*cyclic_test[:,1:2]/6)

sin_hourt    = np.sin(2*np.pi*cyclic_test[:,2:3]/24)
cos_hourt    = np.cos(2*np.pi*cyclic_test[:,2:3]/24)

sin_wintert    = np.sin(2*np.pi*cyclic_test[:,3:4]/4)
cos_wintert    = np.cos(2*np.pi*cyclic_test[:,3:4]/4)

sin_springt    = np.sin(2*np.pi*cyclic_test[:,4:5]/4)
cos_springt    = np.cos(2*np.pi*cyclic_test[:,4:5]/4)

sin_summert    = np.sin(2*np.pi*cyclic_test[:,5:6]/4)
cos_summert    = np.cos(2*np.pi*cyclic_test[:,5:6]/4)

sin_fallt    = np.sin(2*np.pi*cyclic_test[:,6:7]/4)
cos_fallt    = np.cos(2*np.pi*cyclic_test[:,6:7]/4)

sin_year_dayt    = np.sin(2*np.pi*cyclic_test[:,7:8]/365)
cos_year_dayt    = np.cos(2*np.pi*cyclic_test[:,7:8]/365)

test_cyclic = np.concatenate((sin_montht, cos_montht,
                               sin_weekt, cos_weekt,
                               sin_hourt,cos_hourt,
                               sin_wintert,cos_wintert,
                               sin_springt,cos_springt,
                               sin_summert ,cos_summert,

sin_fallt,cos_fallt,sin_year_dayt,cos_year_dayt ), axis=1)

test_cyclic.shape
```

```
(12130, 16)
```

```
test = np.concatenate((test_numerical,test_categorical, test_cyclic),
axis=1)
test.shape
```

```
(12130, 21)
```

```
test_df = pd.DataFrame(data = test.transpose(), index = ['aep',

'Is_holiday1','Is_holiday2',

'Is_Weekend1','Is_Weekend2',

'sin_month', 'cos_month',

'sin_week','cos_week',
                                                           'sin_hour',
'cos_hour',

'sin_wintert','cos_wintert',

'sin_springt','cos_springt',

'sin_summert' ,'cos_summert',

'sin_fallt','cos_fallt',

'sin_year_dayt','cos_year_dayt']).transpose()

test_df.to_csv('9_AEP_test.csv', index=False)

test_df.shape
```

```
(12130, 21)
```

```
4*((24*24)+24^2+24)
```

```
2312
```

```
(84907)/(10*(24+24))
```

```
176.88958333333332
```

```
84907
```

```
84907
```